Complexity of redundancy detection on RDF graphs in the presence of rules, constraints, and queries

Editor(s): Diego Calvanese, Free University of Bozen-Bolzano, Italy; Thomas Lukasiewicz, University of Oxford, UK
Solicited review(s): Adila A. Krisnadhi, Kno.e.sis Center, Wright State University, USA; one anonymous reviewer

Reinhard Pichler [a], Axel Polleres [b,c], Sebastian Skritek [a,*] and Stefan Woltran [a,**]
[a] *Vienna University of Technology, Faculty of Informatics, Favoritenstraße 9, A-1040 Wien, Austria*
E-mail: {lastname}@dbai.tuwien.ac.at
[b] *Siemens AG Österreich, Siemensstrasse 90, A-1210 Wien, Austria*
E-mail: axel.polleres@siemens.com
[c] *Digital Enterprise Research Institute, National University of Ireland, Galwway, IDA Business Park, Lower Dangan, Ireland*

Abstract. Based on practical observations on rule-based inference on RDF data, we study the problem of redundancy detection on RDF graphs in the presence of rules (in the form of Datalog rules) and constraints, (in the form of so-called tuple-generating dependencies), and with respect to queries (ranging from conjunctive queries up to more complex ones, particularly covering features of SPARQL, such as union, negation, or filters). To this end, we investigate the influence of several problem parameters (like restrictions on the size of the rules, the constraints, and/or the queries) on the complexity of detecting redundancy. The main result of this paper is a fine-grained complexity analysis of both graph and rule minimisation in various settings.

Keywords: RDF, Optimisation, Rules, Constraints, Computational Complexity

1. Introduction

The Semantic Web promises to enable computers to gather machine readable meta-data in the form of RDF statements published on the Web and make inferences about these statements by means of accompanying standards such as RDFS and OWL2. While complete OWL2 reasoning is hard – and in many cases even inappropriate for Web data [15] – (incomplete) rule-based inference is becoming quite popular and supported by many RDF stores and query engines: frameworks like GiaBATA [17], Jena, Sesame, OWLIM,[1] etc. allow for custom inference on top of RDF stores, supporting different rule-based fragments of RDFS and OWL. Several such fragments have been defined in the literature, such as ρDF [22], DLP [11], OWL$^-$ [7], ter Horst's pD* [28], or SAOR [16], and – more recently – the W3C standardised OWL2RL, a fragment of OWL implementable purely in terms of rule-based inference [20]. All these fragments have in common that they are implementable by simple Datalog-like rules over RDF. As an example, depicted in Figure 1, let us take (1) the sub-property rule from RDFS [14, Section 7.3, rule rdfs7], rules (2)–(6) from OWL2RL [14, Section 4.3, rules prp-inv1,prp-

*Corresponding author. E-mail: skritek@dbai.tuwien.ac.at.
**A preliminary version of this paper appeared at RR2010 [26].

[1] cf. `http://jena.sourceforge.net/`, `http://openrdf.org/`, and `http://ontotext.com/owlim/`

(1) { S P O . P *subPropertyOf* Q . uri(Q) } $\Rightarrow$ { S Q O }
(2) { S P O . P *inverseOf* Q . uri(O) $\wedge$ uri(Q) } $\Rightarrow$ { O Q S }
(3) { S P O . P *inverseOf* Q . blank(O) $\wedge$ uri(Q) } $\Rightarrow$ { O Q S }
(4) { S P O . P *type SymmetricProperty* . uri(O) } $\Rightarrow$ { O P S }
(5) { S P O . P *type SymmetricProperty* . blank(O) } $\Rightarrow$ { O P S }
(6) { S P_0 O_1. ... O_n P_n O. P *propertyChainAxiom* (P_0 ...P_n) } $\Rightarrow$ { S P O }

(7) G_D = { <http://semanticweb.org/wiki/Pat_Hayes> *made* <http://www.w3.org/TR/rdf-mt/>.
(8) <http://semanticweb.org/wiki/Pat_Hayes> *name* "Patrick J. Hayes".
(9) <http://www.w3.org/TR/rdf-mt/> *creator* "Patrick J. Hayes". }

(10) G_O = { *name subPropertyOf label*.
(11) *inverseOf type SymmetricProperty*.
(12) *made inverseOf maker*.
(13) *maker inverseOf made*.
(14) *creator propertyChainAxiom* (*maker label*) . }

Fig. 1. Rules and RDF graph for the example: Rule (1) is taken from RDFS, rules (2)–(6) from OWL2RL. The RDF graph G_D (7)–(9) describes authors and their publications, and G_O (10)–(14) the ontology used by G_D.

symp,prp-spo2] representing inverse properties, symmetric properties, and property chains:[2]

Further, let G_D (Figure 1, (7)–(9)) be an RDF graph talking about authors and their publications. Moreover, let graph G_O (Figure 1, (10)–(14)) be part of the ontology defining the terms used in G_D.

When storing the graph $G = G_D \cup G_O$ in an RDF store that supports inference over rules (1)–(6), different questions of redundancy arise like if some statements may be deleted since they can be inferred by the rules. In our example, e.g. statement (9) as well as statement (13) may be deleted, since they could be reproduced by inference. Similarly, suppose that we transfer the graph $G = G_D \cup G_O$ to a "weaker" RDF store that only supports rules (1)–(3). Then the question is if we thus loose any inferences. In fact, the answer is no.

We emphasize that the investigations in this paper are very much driven by practical observations on published Linked Data and common inference rules: redundancies do occur in practice, both in terms of published sets of inference rules for RDF as well as within published data from popular Linked Data datasets. For instance, on the one hand, standard rule sets such as OWL2RL are known to be non-minimal [20, Section 4.3]. On the other hand, if we take as an example the RDF data published at `http://dbpedia.org/resource/Vienna` – which is RDF data extracted automatically from Wikipedia within the DBpedia project [5] – we can observe that this data contains a number of redundant RDF triples such as the triple[3]

```
@prefix : <http://dbpedia.org/resource/>
 :Vienna rdfs:label "Vienna" .
```

which is already implied by the RDF Schema semantics via the two triples

```
@prefix : <http://dbpedia.org/resource/>
:Vienna foaf:name "Vienna" .
foaf:name rdfs:subPropertyOf rdfs:label .
```

which can be found on DBpedia itself and within the FOAF ontology, respectively.

Many more such cases of redundancy can be found in published Linked Data online which is why it seems worthwhile to investigate how expensive removing such redundancies would be.

We thus want to be able to answer the general question about redundancy of both triples and rules. How-

[2]We disregard full URIs for common RDF terms, i.e., we just write e.g. *inverseOf*, for <http://www.w3.org/2002/07/owl#inverseOf>, *name* for <http://xmlns.com/foaf/0.1/name>, or *creator* for <http://purl.org/dc/elements/1.1/creator>, etc. Further, $(P_1 \dots P_n)$ in RDF is short for a fresh variable X plus additional triples X *first* P_1 . X_1 *rest* X_2 . $\dots X_n$ *first* P_n . X_n *rest nil* . using reserved terms *first*, *rest*, *nil*.

[3]We use here common Turtle [3] syntax with prefixes to be found at `http://prefix.cc`.

ever, it is often important to limit the minimisation of RDF graphs in such a way that certain consistency conditions must be preserved. These consistency conditions can be expressed by means of constraints [18]. We shall restrict ourselves here to constraints in the form of so-called *tuple-generating dependency (tgd) constraints* [4], which are a generalisation of the familiar foreign-key dependencies in the relational database world. Roughly speaking, a tgd may be viewed as a generalised rule "read" as constraint. I.e. instead of allowing to infer new information, a constraint tests if the current data satisfies certain requirements (cf. [21] for a discussion on the difference between rules and constraints). So, for instance, if we read rules (4)-(5) as constraints, we could say that graph G alone without rules satisfies these constraints, and likewise the closure of G with respect to rules (1)-(3) does. Tgd constraints can be more general than (Horn) rules in that they also allow otherwise unbound, existential variables in the head, possibly occurring in a larger conjunct. That is, tgds are – rather than rules – constraining queries (in the head) "triggered" by bindings coming from a query in the body; for instance, a constraint

(15) { A $made$ D } $\Rightarrow$ { A $label$ N . D $creator$ N}

would hold only on graphs where everybody who made something also has a declared label and that label is also used to denote the creator. Note that constraint (15) holds on the closure of G with respect to rule (1) but – as opposed to the constraint reading of (4)-(5) – not on G alone.

Next, we are interested in redundancy with respect to queries. This might be particularly relevant for RDF stores that expose a narrow SPARQL query interface. For instance, suppose that, in our example, we are interested only in completeness with respect to the query "`SELECT ?D ?L { ?D` *maker* `?M . ?M` *label* `?L }`" which is the SPARQL way of writing a conjunctive query:

(16) { D $maker$ M . M $label$ L } $\to$ $ans(D, L)$

In such a setting, rules (3)–(6) as well as triples (9),(11),(13), and (14) can be dropped. Such redundancy elimination is not unique; for instance, keeping triples (11), (13), and rule (4) we could drop (12), still preserving completeness.

The primary goal of our work is a systematic complexity analysis of both graph and rule minimisation under constraints, as well as with respect to queries. To this end, we investigate the influence of several problem parameters (like restrictions on the size of the rules, constraints, and queries) on the complexity of detecting redundancy. A first important step in this investigation has been recently made by Meier [19]. He studied the following problem: Given a graph G, a set $\mathcal{R}$ of rules and a set $\mathcal{C}$ of tgds, can G be reduced to a proper subgraph $G' \subset G$, such that G' still satisfies $\mathcal{C}$ and the closure of G' under $\mathcal{R}$ coincides with the closure of G under $\mathcal{R}$? For the special case that both the rules in $\mathcal{R}$ and the constraints in $\mathcal{C}$ have bounded size (referred to as *b-boundedness*), this problem was shown to be NP-complete in [19]. In this paper, we want to extend the work initiated in [19] and provide a much more fine-grained analysis of the complexity, e.g., by weakening or strengthening restrictions such as b-boundedness and by considering redundancy elimination that only preserves RDF *entailment* (rather than keeping the closure of the original graph under the original rules unchanged) and additionally considering redundancy with respect to queries.

We shall come up with a collection of complexity results, ranging from tractability to Σ_3^P-completeness. Additionally, we address the orthogonal problems of rule minimisation and the problem of reducing rules or triples without preserving completeness of the entire closure, but only ensuring that the answers to certain queries are preserved.

We shall also discuss further variations of the graph and rule minimisation problem. For instance, the rules and tgds in [19] do not allow variables in predicate positions, which is a severe restriction in the sense that many of the common RDF inferences rules are not covered (e.g., all except rules (4) and (5) above). We will not make this restriction, since it can be dropped without significant change of the complexity results.

Organisation of the paper and summary of results. In Section 2, we recall some basic notions and results. A conclusion and an outlook to future work are given in Section 7. Sections 3–6 contain the main results of the paper, namely:

- *Graph Minimisation.* In Section 3, we provide a comprehensive complexity analysis of the RDF graph minimisation problem, both when full reconstruction of the graph or only RDF entailment is required. We study various settings which result from different restrictions on the rules and/or tgds like restricting their size, considering them as fixed, omitting them, or imposing no restrictions at all. Our complexity results range from tractability to Σ_3^P-completeness.

- *Rule Minimisation.* In Section 4, we consider the problem of minimising the set of rules. We show that the problem of finding redundant rules with respect to a given RDF graph is NP-complete for b-bounded rules and not harder than Δ_2^P for arbitrary rules. Note that rule minimisation is closely related to the field of Datalog equivalence and optimisation. We therefore discuss how the large body of results in this area can be fruitfully applied to the problems studied here.

• *Graph Minimisation w.r.t. Queries.* In Section 5, we study how guaranteeing completeness only w.r.t. a given set of conjunctive queries (CQs) or unions of conjunctive queries (UCQs) influences the complexity for each of the above settings. Considering different restrictions on the size of the queries, hardness never exceeds Σ_3^P, but for some settings raises by two levels in the polynomial hierarchy compared to Section 3. Finally we extend our findings to the problem of rule minimisation. We shall also briefly touch on full SPARQL queries beyond unions of conjunctive queries.

• *Problem Variations.* In Section 6, we analyze the complexity of further problems which are either variations of or strongly related to the graph and rule minimisation problems mentioned above. For instance, rather than asking if an RDF graph contains redundant tuples, we consider the problem whether an RDF graph can be reduced below a certain size. We show that this problem is NP-complete also in those settings where the graph minimisation problem is tractable. We also discuss the effect of allowing blank nodes in predicate positions in the Datalog rules.

The present paper significantly extends the conference version [26] in several ways: above all, full proofs of all results are given here. In Section 4 and Section 5, all of the $\Delta_2^P[\log n]$-completeness results are new (note that in the conference version, the Δ_2^P-membership was shown while hardness was only proved for NP- or coNP). Section 5 now also considers for some settings the extension to UCQs with safe negation. Further, in Section 5 we initiate the study of the influence of SPARQL as query language for the graph minimisation problem. Finally, the proof sketches of hardness proofs in [26] were based on a reduction from the SAT or QSAT problem. In this paper, all hardness proofs except Theorem 6.1 have been rewritten to reduce from graph colorability problems instead of SAT or QSAT to provide much more intuitive reductions.

2. Preliminaries

This section reviews the basic notions of RDF and and RDF Entailment, formally defines the versions of RDF Rules, constraints, and queries considered in this paper, introduces further basic notions and fixes some notational conventions.

2.1. RDF

Let U, B, and L denote pairwise disjoint alphabets for *URI references*, *Blank nodes* (or variables) and *Literals*, respectively. Throughout this paper, unions of these sets are simply denoted by concatenating their names.[4] An RDF statement (or *triple*) is a statement of the form $(s, p, o) \in \mathit{UB} \times U \times \mathit{UBL}$, and an RDF *graph* is a set of triples. In this paper, no distinction is made between variables and blank nodes. Just note that blank nodes/variables appearing in the data are understood to be existentially quantified within the scope of the whole RDF graph they appear in. Elements from B (U) are written as alphanumeric strings starting with an upper case letter (lower case letter or number), elements from L as quoted strings, and – inspired by the common Turtle [3] syntax – RDF statements as white-space separated triples and RDF graphs as '.' separated lists of triples in curly braces. Additionally, the aforementioned shortcut notation for lists is used. That is, graph G_D from above would be written as:

$\{$ $\mathit{patHayes}$ made rdfmt.
$\mathit{patHayes}$ name "Patrick J. Hayes".
rdfmt $\mathit{creator}$ "Patrick J. Hayes" $\}$

A *homomorphism* h between two RDF graphs G_1 and G_2 (written as $h\colon G_1 \to G_2$) is a blank node mapping $h\colon B \to \mathit{UBL}$ such that $h(G_1) \subseteq G_2$, where $h(G_1)$ denotes the graph obtained by replacing in G_1 every variable $\mathtt{B} \in B$ with $h(\mathtt{B})$. A homomorphism h' is an *extension* of a homomorphism h if $h'(\mathtt{B}) = h(\mathtt{B})$ for all blank nodes $\mathtt{B}$ on which h is defined.

It is convenient to define the notion of *entailment* between two RDF graphs via the interpolation lemma from [14, Section 2] rather than in a model-theoretic way: an RDF graph G_1 *entails* G_2, written $G_1 \models G_2$ if a subgraph of G_1 is an instance of G_2, that is, if there exists a homomorphism $h\colon G_2 \to G_1$. Given G_1, G_2, deciding whether there exists a homomorphism $G_2 \to G_1$ (thus also $G_1 \models G_2$) is well known to be NP-complete.

[4]In this paper, a slightly simplified notion of RDF compared to [14] is used, e.g. not considering typed literals separately.

2.2. Rules and constraints

A *basic graph pattern* (BGP) is a set of generalised triples $(s', p', o') \in UBL \times UBL \times UBL$ and a *filter condition* is a conjunct of the unary predicates `uri`$(\cdot)$, `blank`$(\cdot)$, `literal`$(\cdot)$ (denoting the unary relations U, B, and L, respectively). A *filtered basic graph pattern* (FBGP) is a BGP conjoined with a filter condition, the latter containing only variables already appearing in the BGP. Given an FBGP P, denote its components with $BGP(P)$ and $F(P)$, that is its BGP and its filter condition, respectively. Homomorphisms between BGPs and from BGPs to RDF graphs are defined analogously as before. Hence a homomorphism $h\colon P_1 \to P_2$ is a blank node mapping $h\colon B \to UBL$ such that $h(P_1) \subseteq P_2$, where P_1 is a BGP and P_2 is either a BGP or an RDF graph. Further, a blank node mapping h is a homomorphism $h\colon P_1 \to P_2$ from an FBGP P_1 into a BGP or RDF graph P_2 if h is a homomorphism $h\colon BGP(P_1) \to P_2$, and $h(F(P_1))$ is satisfied. Thereby $h(F(P_1))$ is satisfied if for every filter condition `filter(x)` in $F(P_1)$ (where `filter` $\in$ $\{$`uri`, `blank`, `literal`$\}$) the value of $\hat{h}(x)$ is of the correct type where $\hat{h}(x) = h(x)$ if $x \in B$ and $\hat{h}(x) = x$ for $x \in UL$.

Note that for a FBGP P, given a homomorphism h on BGP(P) it is easy to check if h also satisfies F(P) (a more detailed discussion of how to do this can be found in Section 6.3). Therefore, for a (F)BGP P, an RDF graph or BGP G and a blank node mapping h, throughout the paper only $h(P) \subseteq G$ is used to denote the property that h is indeed a homomorphism $h\colon P \to G$. Hence it is not distinguished if P is a BGP or FBGP, and in the latter case the additional requirement that h satisfies F(P) is not stated explicitly.

The type of constraints considered in this paper are *RDF tuple-generating dependency (tgd) constraints* (cf. [4]). A tgd constraint (or simply constraint) r is defined as $\mathcal{A}nte \Rightarrow \mathcal{C}on$, where the *antecedent* $\mathcal{A}nte$ is a FBGP and the *consequent* $\mathcal{C}on$ is a BGP. A constraint $\mathcal{A}nte \Rightarrow \mathcal{C}on$ is a short-hand notation for the first-order formula $\forall \vec{X}\big(\mathcal{A}nte(\vec{X}) \rightarrow (\exists \vec{Y})\mathcal{C}on(\vec{X}, \vec{Y})\big)$ (where $\vec{Y}$ denotes the blank nodes occurring in $\mathcal{C}on$ only, while $\vec{X}$ are the remaining blank nodes). Hence, a constraint $\mathcal{A}nte \Rightarrow \mathcal{C}on$ is satisfied over an RDF graph G if for each homomorphism $h\colon \mathcal{A}nte \to G$ on $\vec{X}$ there exists an an extension h' of h to $\vec{Y}$, s.t. $h'(\mathcal{C}on) \subseteq G$. To increase the readability, sometimes the quantifiers and variable vectors will be stated explicitly.

RDF rules (or simply rules), are syntactically restricted constraints, where $\vec{Y} = \emptyset$, i.e. all variables appearing in $\mathcal{C}on$ also appear in $\mathcal{A}nte$ (akin to the common notion of safety [29] in Datalog). In the following, RDF rules with an empty filter condition will be called *Datalog rules*.[5] Given a set $\mathcal{R}$ of rules and an RDF graph G, the *closure* of G with respect to $\mathcal{R}$, written $Cl_{\mathcal{R}}(G)$, is defined as usual by the least fix-point of the immediate consequence operator. I.e. using the notation from [17], for a rule $r \in \mathcal{R}$ with $r\colon \mathcal{A}nte \Rightarrow \mathcal{C}on$, let $T_r(G) = \{\mu(\mathcal{C}on) \mid \mu\colon \mathcal{A}nte \to G\}$. Accordingly, let $T_{\mathcal{R}}(G) = \bigcup_{r \in \mathcal{R}} T_r(G)$. Also, let $G_0 = G$ and $G_{i+1} = G_i \cup T_{\mathcal{R}}(G_i)$ for $i \geq 0$. Then, there exists a smallest n such that $G_{n+1} = G_n$ and $Cl_{\mathcal{R}}(G) = G_n$.

A rule or constraint is said to be *b-bounded* if both, antecedent and consequent contain at most b triples. Given an RDF graph and an arbitrary tgd τ, testing if G satisfies τ is Π_2^P-hard [10, Proposition 5.5, (1)]. On the other hand, for b-bounded tgds, it follows from well-known results that the problem is tractable, cf. [19, Proposition 3]. Testing if an RDF rule is applicable is well-known to be NP-complete, and tractable in case of b-bounded rules (cf. [19, Proposition 9]).

For a given graph G or a given set $\mathcal{R}$ of rules, let X_G and $X_{\mathcal{R}}$ ($X \in \{U, B, L\}$) denote the subset of U (resp. B, L) used in G or $\mathcal{R}$, respectively.

2.3. Queries

2.3.1. Conjunctive Queries

A *conjunctive query (CQ)* q over an RDF graph G is of the form $G_q \to ans(\vec{X})$, where G_q is a FBGP, ans is a distinguished predicate, and $\vec{X}$ is a vector of blank nodes occurring in G_q. G_q is referred to as the *body* of q ($body(q)$), and $ans(\vec{X})$ as the *head* of q ($head(q)$). A *union of conjunctive queries (UCQs)* is a set of CQs, all having the same head. The result of a CQ q over some RDF graph G is defined as the set $q(G) = \{\tau(\vec{X}) \mid \tau\colon body(q) \to G\}$. The result of a UCQ is the union of the results of its CQs.

For extending (U)CQs to *(U)CQs$^\neg$* (i.e. allowing for negation), given a query q, partition BGP($body(q)$) into two sets, $pos(q)$ and $neg(q)$. Intuitively, $pos(q)$ encodes positive information that must be satisfied by G, while G must not contain the negative information encoded by $neg(q)$. In the following, assume all

[5]In fact, in most parts of this paper, only Datalog rules will be considered. Extension to arbitrary RDF rules will be revisited at the end of Section 6, concluding that this extension does not change any of the results presented.

(U)CQs$^{\neg}$ to be safe, that is, all blank nodes appearing in $head(q)$ and $neg(q)$ must also appear in $pos(q)$. The result of a CQ$^{\neg}$ q over some RDF graph G is defined as the set $q(G) = \{\tau(\vec{X}) \mid \tau\colon pos(q) \cup F(body(q)) \to G$ and $\tau(t) \notin G$ for all $t \in neg(q)\}$. The result of a UCQ$^{\neg}$ is the union of the results of its CQs$^{\neg}$. In the following, when considering a CQ or UCQ q (i.e. $neg(q) = \emptyset$), the expressions $body(q)$ and $pos(q)$ are used interchangeably.

A conjunctive query q is said to be *body-b-bounded* if $body(q)$ contains at most b triples. Moreover, q is said to be *head-b-bounded* if $|\vec{X}| \leq b$ for some constant b (however, $body(q)$ may be arbitrary). A set $\mathcal{Q}$ of (U)CQs is body-b-bounded (resp. head-b-bounded) if every $q \in \mathcal{Q}$ is body-b-bounded (resp. head b-bounded).

2.3.2. SPARQL graph patterns

SPARQL [27] is the W3C Recommendation for a query language for RDF. A SPARQL query consists of two main parts. In the body of a query, a set of variable bindings is created. The head of the query then allows to apply several solution modifiers (like projection, order, ...) on these variable mappings. The main part of the body are the so called SPARQL *graph patterns*, which are used to define the variable bindings. They have been studied in [24]. Following [24], this paper concentrates on SPARQL graph patterns only.

Assume a set V of variables disjoint from UBL. The basic building block of a SPARQL graph pattern is a SPARQL triple pattern, which is a triple in $VU \times VU \times VUL$. Note that in this work, SPARQL triple patterns are not allowed to contain blank nodes. Therefore elements from V are written as alphanumeric strings starting with an upper case letter just as elements from B, since the specific kind of an element will be always clear from the context.

Complex SPARQL graph patterns are constructed from SPARQL triple patterns by using operators AND, OPT, UNION, and FILTER. Formally, SPARQL graph patterns are recursively defined as follows. (1) A SPARQL triple pattern is a SPARQL graph pattern, and (2) if P_1 and P_2 are SPARQL graph patterns and R is a SPARQL filter expression, then $(P_1 \text{ AND } P_2)$, $(P_1 \text{ OPT } P_2)$, $(P_1 \text{ UNION } P_2)$, and $(P_1 \text{ FILTER } R)$ are SPARQL graph patterns. Thereby a SPARQL filter expression is built of elements from ULV, logical connectives, equality and inequality symbols, predicates similar to the filter conditions `uri`(.) and `literal`(.) introduced above, and further features (for a complete reference see [27]).

For a SPARQL triple pattern t, let $vars(t)$ denote the set of variables occurring in t, and for a SPARQL graph pattern P let $vars(P)$ denote the set of variables that occur in the triples that compose P.

Next, the semantics of SPARQL graph patterns is defined, following closely the definitions proposed in [24]. A mapping μ is a partial function $\mu : V \to U$. The domain of μ, denoted by $dom(\mu)$, is the set of all variables from V for which μ is defined. Given a triple pattern t and a mapping μ such that $vars(t) \subseteq dom(\mu)$, denote by $\mu(t)$ the RDF triple obtained by replacing the variables in t according to μ. Two mappings μ_1 and μ_2 are said to be *compatible*, denoted by $\mu_1 \sim \mu_2$, if for every $?X \in dom(\mu_1) \cap dom(\mu_2)$ it holds that $\mu_1(X) = \mu_2(X)$. The mapping with empty domain, denoted by $\mu_\emptyset$ is compatible with any mapping. The fact that a mapping μ satisfies a SPARQL filter expression R is denoted as $\mu \models R$. Since SPARQL filter expressions are of little importance in this paper, a formal definition of the semantics of SPARQL filter expression is omitted. Note however that their formal semantics represents their intuitive meaning. For details, see [24].

The semantics of SPARQL graph patterns is defined with the help of the following operations between sets of mappings that resemble relational operators over sets of tuples. Let M_1 and M_2 be sets of mappings. Then the *join* and the *left-outer join* between M_1 and M_2 are defiend as follows:

$$M_1 \bowtie M_2 = \{\mu_1 \cup \mu_2 \mid \mu_1 \in M_1, \mu_2 \in M_2 \text{ and } \mu_1 \sim \mu_2\}$$

$$M_1 \quad M_2 = (M_1 \bowtie M_2) \cup \{\mu \in M_1 \mid \forall \mu' \in M_2 : \mu \not\sim \mu'\}$$

This allows one to formalize the evaluation of a SPARQL graph pattern over an RDF graph G as a function $[\![\cdot]\!]_G$ that, given a SPARQL graph pattern, returns a set of mappings. Formally, $[\![P]\!]_G$ is defined recursively as follows [24]:

1. If P is a triple pattern t, then $[\![P]\!]_G = \{\mu \mid dom(\mu) = vars(t) \text{ and } \mu(t) \in G\}$.
2. If $P = (P_1 \text{ AND } P_2)$, then $[\![P]\!]_G = [\![P_1]\!]_G \bowtie [\![P_2]\!]_G$.
3. If $P = (P_1 \text{ OPT } P_2)$, then $[\![P]\!]_G = [\![P_1]\!]_G \quad [\![P_2]\!]_G$.
4. If $P = (P_1 \text{ UNION } P_2)$, then $[\![P]\!]_G = [\![P_1]\!]_G \cup [\![P_2]\!]_G$.
5. If $P = (P_1 \text{ FILTER } R)$, then $[\![P]\!]_G = \{\mu \in [\![P_1]\!]_G \mid \mu \models R\}$.

Given a mapping μ, it was shown in [24] that deciding if $\mu \in [\![P]\!]_G$ is PSPACE-complete.

2.4. Graphs and further notation

A *graph* $G = (V, E)$ consists of a set V of nodes (also called vertices) and a set E of edges. Thereby an edge $e \in E$ is a pair (v_i, v_j) of nodes $v_i, v_j \in V$. Throughout the paper, unless stated otherwise, edges are considered to be undirected, i.e. $(v_i, v_j) = (v_j, v_i)$. A graph $G' = (V', E')$ is a subgraph of G if $V' \subseteq V$ and $E' \subseteq E$. A *clique* is a complete graph, i.e. $G = (V, E)$ is a clique if for all pairs $v_i, v_j \in V$, the edge $(v_i, v_j) \in E$. Further, given some graph $G = (V, E)$, a *3-coloring* of V is a mapping $\phi\colon V \to \{1, 2, 3\}$. A 3-coloring ϕ is called a *valid 3-coloring of G* if $\phi(v_i) \neq \phi(v_j)$ for all pairs $v_i, v_j \in V$ s.t. $(v_i, v_j) \in E$. Also recall the well-known NP-complete problems graph 3-colorability (3COL) and vertex cover.

Introducing some additional notation, let $[n]$ denote the set $\{1, \dots, n\}$. Finally, recall that given some set V, subsets $V_1, \dots, V_n \subseteq V$ are a *partition* of V if $V_i \cap V_j = \emptyset$ for $i, j \in [n]$ with $i \neq j$ and $\bigcup_{i=1}^{n} V_i = V$.

2.5. Complexity classes and complete problems

The different problems studied in this paper are shown to be complete for a handful of different complexity classes. In the following, a short recapitulation of the less well-known classes among them is given. Further, the less well-known problems used in the hardness proofs are introduced.

All problems under consideration are shown to be contained in some class of the polynomial hierarchy. Recall that the polynomial hierarchy contains the classes Σ_i^P and Π_i^P for all $i \geq 0$, which are defined as $\Sigma_0^P = \Pi_0^P = \mathrm{P}$, and $\Sigma_{i+1}^P = \mathrm{NP}^{\Sigma_i^P}$ and $\Pi_{i+1}^P = \mathrm{coNP}^{\Sigma_i^P}$ for $i \geq 0$. The prototypical problem complete for the the k^{th} level of the polynomial hierarchy are QSAT$_{\exists,k}$ (for Σ_k^P) and QSAT$_{\forall,k}$ (for Π_k^P) (cf. for example [23, Theorem 17.10]). However, due to the syntactic restriction of RDF to consider only triples, reducing from QSAT to the problems studied in this paper turns out to be rather tedious. Therefore, in most cases, hardness will be shown via reduction from the appropriate quantified variants of 3COL.

Definition 2.1 (Q-3COL$_{\exists,3}$). *Let Q-3COL$_{\exists,3}$ be the following decision problem:*
INPUT: A graph $G = (V, E)$ together with a partition (V_1, V_2, V_3) of V.
QUESTION: Does there exist a 3-coloring ϕ_1 of V_1, such that for all possible 3-colorings ϕ_2 of V_2 there exists a 3-coloring ϕ_3 of V_3, such that the 3-coloring $\phi = \phi_1 \cup \phi_2 \cup \phi_3$ is a valid *3-coloring of G.*

Definition 2.2 (Q-3COL$_{\exists,2}$). *Let Q-3COL$_{\exists,2}$ be the following decision problem:*
INPUT: A graph $G = (V, E)$ together with a partition (V_1, V_2) of V.
QUESTION: Does there exist a 3-coloring ϕ_1 of V_1, such that for all possible 3-colorings ϕ_2 of V_2 the 3-coloring $\phi = \phi_1 \cup \phi_2$ is no valid *3-coloring of G?*

The Σ_3^P-completeness of Q-3COL$_{\exists,3}$ and the Σ_2^P-completeness of Q-3COL$_{\exists,2}$ follow immediately from [1]. There, the property *k-round 3-colorability* of graphs was introduced via a k-round game and shown to be complete for Σ_k^P ([1, Theorem 11.4]). An inspection of the proof of this theorem reveals the completeness results for Q-3COL$_{\exists,3}$ and Q-3COL$_{\exists,2}$.

Next, consider the co-problem of Q-3COL$_{\exists,2}$, which can be defined as follows.

Definition 2.3 (Q-3COL$_{\forall,2}$). *Let Q-3COL$_{\forall,2}$ be the following decision problem:*
INPUT: A graph $G = (V, E)$ together with a partition (V_1, V_2) of V.
QUESTION: For all possible 3-colorings ϕ_1 of V_1, does there exist a 3-coloring ϕ_2 of V_2 such that the 3-coloring $\phi = \phi_1 \cup \phi_2$ is a valid *3-coloring of G?*

The Π_2^P-completeness of the problem follows immediately.

The final complexity class of interest in this paper is the class $\Delta_2^P[\log n]$. It contains all decision problems that can be solved by a deterministic Turing machine having access to an NP-oracle in polynomial time while calling the oracle at most $O(\log n)$ times (where n is the size of the input). The following problem is known to be complete for $\Delta_2^P[\log n]$ [31].

Definition 2.4 (ODD CLIQUE). *Let ODD CLIQUE be the following decision problem:*
INPUT: A graph $G = (V, E)$.
QUESTION: Is the size of the biggest clique in G odd?

In fact, it is sometimes convenient not to prove membership for $\Delta_2^P[\log n]$ directly, but instead to show membership for the class $\mathrm{P}_{\|}^{\mathrm{NP}}$. However, since $\Delta_2^P[\log n]$ is known to be equivalent with the class $\mathrm{P}_{\|}^{\mathrm{NP}}$ (cf. for example [23, Theorem 17.7]), this immediately also gives $\Delta_2^P[\log n]$ membership.

Thereby $\mathrm{P}_{\|}^{\mathrm{NP}}$ is the class of all decision problems that can be solved in polynomial time by a deterministic Turing machine M with an NP-oracle under the following restrictions. M is allowed to call the oracle a polynomial number of times, but all these calls must be performed in parallel. I.e. all these calls must be independent of the results returned from the other calls, and must not be adapted according to the answers from the oracle. Note that it follows immediately that a problem is still in $\Delta_2^P[\log n]$ (hence also in $\mathrm{P}_{\|}^{\mathrm{NP}}$) if it can be solved in polynomial time by the machine M described above, but allowing in addition a fixed number of queries to the oracle that depend of the results retrieved from the polynomial number of parallel calls.

3. RDF graph minimisation

Given an RDF dataset and an additional set of rules that allow one to infer information from the explicitly stored data, the first question of interest is – beside asking what information can be actually inferred – if it is indeed necessary to keep all the explicit data, or if some of this data could be removed because it is still derivable from the remaining data and the rules. Therefore this is the basic question considered in this section. As already motivated in the introduction, beside a set of rules, it might also be the case that certain constraints are defined that must hold on the dataset. This leads to the two problems whose complexity is investigated in this section, formally defined as follows:

Definition 3.1. *Let* MINI-RDF$^{\models}(G, \mathcal{R}, \mathcal{C})$ *be the following decision problem:*
INPUT: RDF graph G, set $\mathcal{R}$ of RDF rules, set $\mathcal{C}$ of tgds (G satisfies $\mathcal{C}$).
QUESTION: Is there a $G' \subset G$ s.t. $Cl_{\mathcal{R}}(G') \models Cl_{\mathcal{R}}(G)$ and G' satisfies $\mathcal{C}$?

Definition 3.2. *Let* MINI-RDF$^{\subseteq}(G, \mathcal{R}, \mathcal{C})$ *be the following decision problem [19]:*
INPUT: RDF graph G, set $\mathcal{R}$ of RDF rules, set $\mathcal{C}$ of tgds (G satisfies $\mathcal{C}$).
QUESTION: Is there a $G' \subset G$ s.t. $Cl_{\mathcal{R}}(G) = Cl_{\mathcal{R}}(G')$ and G' satisfies $\mathcal{C}$?

The main motivation for these two problems (maybe even more important than the obvious goal of reducing the size of the dataset) is the avoidance of redundancy in the stored data. Thereby two kinds of redundancy elimination are addressed by the above settings. On the one hand, in MINI-RDF$^{\subseteq}$, triples which can be restored via the rules are considered as redundant. On the other hand, the minimisation of RDF graphs via entailment is an important topic on RDF in general. Minimisation via entailment allows to replace a graph G by a proper subgraph $\bar{G} \subset G$ if $\bar{G} \models G$ holds, i.e. one checks if G is lean (see [12]). The MINI-RDF$^{\models}(G, \mathcal{R}, \mathcal{C})$ problem introduced above combines these two approaches and thus yields the strongest redundancy criterion. Nevertheless, in most cases, its complexity is not higher than for MINI-RDF$^{\subseteq}$ (see Theorem 3.1).

3.1. Overview of results

It turns out that both problems, MINI-RDF$^{\subseteq}$ and MINI-RDF$^{\models}$ are Σ_3^P-complete in the general case. Thus several restrictions on the input parameters (more concrete on the set of rules and set of constraints) are taken into account. Based on these restrictions, the complexity of the problems varies between tractability and Σ_3^P-completeness. The results are summarised in the following theorem.

Theorem 3.1. *For* MINI-RDF$^{\models}$ *and* MINI-RDF$^{\subseteq}$*, the complexity with respect to different assumptions on the input (arbitrary, b-bounded, or fixed rule set; arbitrary, b-bounded, fixed, or no constraints) is as depicted in Table 1.*

The remainder of this section is devoted to discuss and prove the correctness of the entries in Table 1. First of all, note that several settings are special cases of others. The following lemma makes some of these relationships explicit. It thus justifies that in order to show the correctness of Theorem 3.1 it is not necessary to give an explicit completeness proof for each entry in Table 1, but points out a proof plan for Theorem 3.1.

Lemma 3.2. *The graph in Fig. 2 correctly describes the dependencies between the problems (identified by their line number) in Table 1, i.e.: if there is an arrow from A to B, then B is a special case of A.*

Proof. The Lemma follows immediately from the observation that arbitrary sets $\mathcal{R}$ (resp. $\mathcal{C}$) include b-bounded sets $\mathcal{R}$ (resp. $\mathcal{C}$), which in turn include sets of constant size. Finally, the empty set is of constant size. □

Hence an arrow from A to B means that membership results for A hold also for B, and that hardness results for B apply also to A. Note that the graph in Fig. 2 is not complete, i.e. not all dependencies are made explicit, but only those needed in order to prove Theorem 3.1. To prove this theorem, it now suffices to show membership for both problems for (1) and (2), and additionally for MINI-RDF$^{\subseteq}$ for setting (8). On the other hand, hardness must be shown for both problems for setting (9), in addition for MINI-RDF$^{\models}$ in setting (12)

		MINI-RDF$^{\models}$	MINI-RDF$^{\subseteq}$
(1)	$\mathcal{R}$ arb., $\mathcal{C}$ arb.	Σ_3^P-complete	Σ_3^P-complete
(2)	$\mathcal{R}$ arb., $\mathcal{C}$ bb	NP-complete	NP-complete
(3)	$\mathcal{R}$ arb., $\mathcal{C}$ fixed	NP-complete	NP-complete
(4)	$\mathcal{R}$ arb., $\mathcal{C} = \emptyset$	NP-complete	NP-complete
(5)	$\mathcal{R}$ bb., $\mathcal{C}$ arb.	Σ_3^P-complete	Σ_3^P-complete
(6)	$\mathcal{R}$ bb, $\mathcal{C}$ bb	NP-complete	NP-complete [19]
(7)	$\mathcal{R}$ bb, $\mathcal{C}$ fixed	NP-complete	NP-complete
(8)	$\mathcal{R}$ bb, $\mathcal{C} = \emptyset$	NP-complete	in P
(9)	$\mathcal{R}$ fixed, $\mathcal{C}$ arb.	Σ_3^P-complete	Σ_3^P-complete
(10)	$\mathcal{R}$ fixed, $\mathcal{C}$ bb	NP-complete	NP-complete
(11)	$\mathcal{R}$ fixed, $\mathcal{C}$ fixed	NP-complete	NP-complete
(12)	$\mathcal{R}$ fixed, $\mathcal{C} = \emptyset$	NP-complete	in P

Table 1

The complexity of MINI-RDF$^{\models}$ and MINI-RDF$^{\subseteq}$ w.r.t. input parameters ("bb" indicates the set to be b-bounded, and "arb." allows for arbitrary sets.)

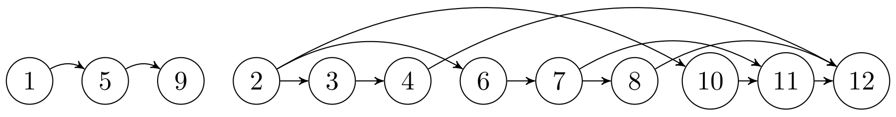

Fig. 2. Dependency graph: Numbers refer to lines in Table 1. An arrow from A to B means that B is a special case of A.

and for MINI-RDF$^{\subseteq}$ for settings (4) and (11). The concrete proofs are given in Section 3.2, but beforehand, the rough intuition of these results is sketched.

The idea of the Σ_3^P-membership in the most general case, (1), can be seen by the following guess and check algorithm that is allowed to call a Π_2^P oracle for the checks. In order to solve the problem, one has to guess: a subgraph G' of G, a sequence of rule applications on G', and for each rule application a homomorphism justifying that the rule is applicable. Note that $Cl_{\mathcal{R}}(G') \subseteq AD^3$ (with $AD = U_G U_{\mathcal{R}} B_G B_{\mathcal{R}} L_G L_{\mathcal{R}}$). Hence if considering all possible rule applications of length $|AD|^3$, one of them has to return $Cl_{\mathcal{R}}(G')$. The most expensive check is to test if G' satisfies $\mathcal{C}$. However, it obviously fits into Π_2^P. The completeness result shows that the above intuition of the sources of complexity for the problem is indeed correct.

Concerning the restricted settings from Theorem 3.1, the reasons for the lower complexity are the following properties: If $\mathcal{R}$ is a b-bounded set, then $Cl_{\mathcal{R}}(G')$ can be computed in polynomial time [19, Proposition 9] and if $\mathcal{C}$ is a b-bounded set, then testing if G' satisfies $\mathcal{C}$ is in PTIME [19, Proposition 3]. These two observations lead to the NP-complete settings. For the tractable cases, note that in addition to the two properties mentioned above, if $\mathcal{C} = \emptyset$, then not all subgraphs of G have to be checked, but only those missing exactly one triple from G. It thus suffices to check only a polynomial number of subgraphs, instead of an exponential number.

3.2. Complexity of RDF graph minimisation

Almost all complexity results presented in this section make use of the following observations. First of all, it is easy to see that the condition $Cl_{\mathcal{R}}(G) = Cl_{\mathcal{R}}(G')$ in Definition 3.2 is equivalent to $G \subseteq Cl_{\mathcal{R}}(G')$. The following lemma shows that similarly, for MINI-RDF$^{\models}$, it is enough to show $Cl_{\mathcal{R}}(G') \models G$ rather than $Cl_{\mathcal{R}}(G') \models Cl_{\mathcal{R}}(G)$.

Lemma 3.3. *Let G_1, G_2 be RDF graphs and $\mathcal{R}$ a set of rules. Then the following equivalence holds: $Cl_{\mathcal{R}}(G_2) \models Cl_{\mathcal{R}}(G_1) \Leftrightarrow Cl_{\mathcal{R}}(G_2) \models G_1$.*

Proof. The "$\Rightarrow$"-direction is trivial. To show the "$\Leftarrow$"-direction, recall that the closure $Cl_{\mathcal{R}}(G)$ of a graph G under a set of rules $\mathcal{R}$ is defined by the least fix-point of the immediate consequence operator.

Let $r \in \mathcal{R}$ with $r = \forall \vec{x}(\mathcal{A}nte(\vec{x}) \to \mathcal{C}on(\vec{x}))$. A single application of the immediate consequence operator T w.r.t. r extends the graph G_1 to the graph $G_1' = G_1 \cup \mathcal{C}on(\lambda(\vec{x}))$, where λ is a mapping on $\vec{x}$ with $\mathcal{A}nte(\lambda(\vec{x})) \subseteq G_1$. It suffices to prove the implication $Cl_{\mathcal{R}}(G_2) \models G_1 \Rightarrow Cl_{\mathcal{R}}(G_2) \models G_1'$. From this, the lemma follows by induction on the number of rule applications when computing the closure $Cl_{\mathcal{R}}(G_1)$.

By assumption, $\mathcal{A}nte(\lambda(\vec{x})) \subseteq G_1$. Moreover, $Cl_{\mathcal{R}}(G_2) \models G_1$ holds. Hence, there exists a mapping η on the blank nodes in G_1, s.t. $\eta(G_1) \subseteq G_2$. In total, for $\sigma = \eta \circ \lambda$, we thus have $\mathcal{A}nte(\sigma(\vec{x})) \subseteq G_2$. Since $Cl_{\mathcal{R}}(G_2)$ is closed under $\mathcal{R}$, the inclusion $\mathcal{C}on(\sigma(\vec{x}))) \subseteq G_2$ holds as well. We claim that η is the desired homomorphism $\eta \colon G_1' \to G_2$. Clearly, we have $\eta(G_1) \subseteq G_2$ by the above considerations. It remains to show that η also maps the triples in $G_1' \setminus G_1$ to G_2. Consider $G_1' \setminus G_1 = \mathcal{C}on(\lambda(\vec{x}))$. By definition, $\sigma = \eta \circ \lambda$. Moreover, $\mathcal{C}on(\sigma(\vec{x})) \subseteq G_2$. Hence, $\eta(\mathcal{C}on(\lambda(\vec{x}))) = \mathcal{C}on(\eta \circ \lambda(\vec{x})) = \mathcal{C}on(\sigma(\vec{x})) \subseteq G_2$. In total, we thus have $\eta(G_1') \subseteq G_2$ and, therefore, $Cl_{\mathcal{R}}(G_2) \models G_1'$. □

3.2.1. Settings complete for Σ_3^P

The idea of why both problems are Σ_3^P-complete in the case of unrestricted constraints was already sketched above. Next, the result is shown formally.

Lemma 3.4. *Both,* MINI-RDF$^{\models}(G, \mathcal{R}, \mathcal{C})$ *and* MINI-RDF$^{\subseteq}(G, \mathcal{R}, \mathcal{C})$*, for arbitrary sets $\mathcal{R}$ and $\mathcal{C}$ can be solved in Σ_3^P.*

Proof. First of all, note that every rule with more than one triple in the consequent can be replaced in a preprocessing step by several rules with the same antecedent, each of them containing in its consequent exactly one triple from the original consequent. Further, let r_0 encode some "do nothing" rule $r_0 \colon \emptyset \Rightarrow \emptyset$ and denote with AD the active domain $AD = U_G \cup U_{\mathcal{R}} \cup B_G \cup B_{\mathcal{R}} \cup L_G \cup L_{\mathcal{R}}$. Then the following nondeterministic algorithm decides MINI-RDF$^{\models}(G, \mathcal{R}, \mathcal{C})$ in polynomial time using a Π_2^P-oracle:

```
1. Guess a subgraph $G' \subset G$ with $G' \neq \emptyset$
2. Check (by a $\Pi_2^P$-oracle) if $G'$ satisfies $\mathcal{C}$
                                   If it does not, return "no".
3. Set $\hat{G}_0 = G'$ and $k = |AD|^3$
4. For $j = 1, \ldots, k$:
   (a) Guess a rule $r_i \colon G_i \Rightarrow \{t_i\}$ from $\mathcal{R} \cup \{r_0\}$
   (b) Guess a mapping $\mu \colon G_i \to \hat{G}_{j-1}$ (i.e. from
       the rule body $G_i$ to the intermediate graph)
   (c) Check whether $\mu$ is a homomorphism $G_i \to \hat{G}_{j-1}$. If it is not return "no"
   (d) Set $\hat{G}_j = \hat{G}_{j-1} \cup \{\mu(t_i)\}$
5. Guess a mapping $\lambda \colon G \to \hat{G}_k$
6. If $\lambda$ is a homomorphism $G \to \hat{G}_k$ return "yes",
                                   otherwise return "no"
```

To see that this algorithm indeed runs in polynomial time, observe the following: Step (2) can be easily solved by a Π_2^P-oracle, see [10, Proposition 5.5, (1)]. The remaining steps obviously fit into polynomial time. In (1),(4a),(4b),(5) a polynomial size subgraph respectively mapping is guessed. Finally, testing in steps (3),(4c),(5) if those mappings guessed are homomorphisms is obviously in polynomial time. This concludes the proof for MINI-RDF$^{\models}(G, \mathcal{R}, \mathcal{C})$.

For MINI-RDF$^{\subseteq}(G, \mathcal{R}, \mathcal{C})$, in steps (4) and (5), instead of testing $Cl_{\mathcal{R}}(G') \models G$, one needs to test whether $G \subseteq Cl_{\mathcal{R}}(G')$, which is obviously not harder. □

Hardness is shown in Lemma 3.5 below by reduction from Q-3COL$_{\exists,3}$. Recall the algorithm sketched in the Σ_3^P-membership proof. Then the intuition of the reduction can be described according to the steps in this algorithm. "Guessing a subgraph" corresponds to fixing a coloring on the first set of nodes, while testing if all constraints are satisfied on this subgraph corresponds to testing if for all colorings on the second set of nodes (encoded in the – universally quantified – antecedent of a tgd) there exists a coloring on the third set of nodes such that the combined coloring gives a valid 3-coloring of the graph (both encoded in the – existentially quanitfied – consequent of a tgd). Hence there exists one tgd whose size is linear in the size of the input graph.

Lemma 3.5. *The problems* MINI-RDF$^{\models}(G, \mathcal{R}, \mathcal{C})$ *and* MINI-RDF$^{\subseteq}(G, \mathcal{R}, \mathcal{C})$*, for fixed $\mathcal{R}$ and arbitrary $\mathcal{C}$, are Σ_3^P-hard.*

Proof. Consider the MINI-RDF$^{\subseteq}$ problem first. The hardness is shown by reduction from Q-3COL$_{\exists,3}$. Hence let $\mathcal{G} = ((V, E), (V_1, V_2, V_3))$ be an arbitrary instance of Q-3COL$_{\exists,3}$ with $V = \{v_1, \ldots, v_n\}$. Define an instance $(G, \mathcal{R}, \mathcal{C})$ of MINI-RDF$^{\subseteq}$ as follows. Let $G = G_{cols} \cup G_{v1} \cup G_{col1} \cup G_{col2} \cup G_{e1} \cup G_{e2} \cup G_{neg}$ where

$$G_{cols} = \{0 \ iscol \ 0 \,.\, 1 \ iscol \ 1 \,.\, 2 \ iscol \ 2\}$$
$$G_{v1} \ = \{v_i \ v \ v_i \mid v_i \in V_1\},$$
$$G_{col1} = \{v_i \ a \ 0 \,.\, v_i \ a \ 1 \,.\, v_i \ a \ 2 \mid v_i \in V_1\},$$

$$
\begin{aligned}
G_{col2} &= \{v_i\ c\ 0\ .\ v_i\ c\ 1\ .\ v_i\ c\ 2 \mid v_i \in V_2 \cup V_3\},\\
G_{e1} &= \{0\ e\ 0\ .\ 1\ e\ 1\ .\ 2\ e\ 2\},\\
G_{e2} &= \{0\ e\ 2\ .\ 0\ e\ 1\ .\ 2\ e\ 0\ .\ 2\ e\ 1\ .\ 1\ e\ 2\ .\ 1\ e\ 0\},\\
G_{neq} &= \{0\ neq\ 2\ .\ 0\ neq\ 1\ .\ 2\ neq\ 0\ .\ 2\ neq\ 1\ .\\
&\qquad 1\ neq\ 2\ .\ 1\ neq\ 0\},
\end{aligned}
$$

and each v_i is a new URI for every $v_i \in V$ (by slight abuse of notation, v_i is used to denote both, nodes in V and URIs in G). Next, let the set $\mathcal{R}$ of rules be defined as $\mathcal{R} = \mathcal{R}_{col1} \cup \mathcal{R}_{e1}$, with

$$
\begin{aligned}
\mathcal{R}_{col1} &= \{\{X\ a\ C\ .\ Y\ iscol\ Y\} \Rightarrow \{X\ a\ Y\}\},\\
\mathcal{R}_{e1} &= \{\{Y\ iscol\ Y\} \Rightarrow \{Y\ e\ Y\}\}.
\end{aligned}
$$

Finally, the set $\mathcal{C}$ of constraints is defined as $\mathcal{C} = \mathcal{C}_{col1} \cup \mathcal{C}_0 \cup \mathcal{C}_{\mathcal{G}}$, where

$$
\begin{aligned}
\mathcal{C}_{col1} &= \{\{X\ a\ C\ .\ X\ a\ D\ .\ C\ neq\ D\} \Rightarrow \{0\ e\ 0\}\}\\
\mathcal{C}_0 &= \{\{X\ e\ X\ .\ Y\ iscol\ Y\} \Rightarrow \{Y\ e\ Y\};\\
&\qquad \{X\ e\ X\ .\ Y\ v\ Y\ .\ Z\ iscol\ Z\} \Rightarrow \{Y\ a\ Z\}\}\\
\mathcal{C}_{\mathcal{G}} &= \{G_1 \cup G_2 \Rightarrow G_3 \cup G_4\} \text{ where}\\
&\qquad G_1 = \{v_i\ a\ C_i \mid v_i \in V_1\},\\
&\qquad G_2 = \{v_i\ c\ C_i \mid v_i \in V_2\},\\
&\qquad G_3 = \{v_i\ c\ C_i \mid v_i \in V_3\}, \text{ and}\\
&\qquad G_4 = \{C_i\ e\ C_j \mid (v_i, v_j) \in E\}.
\end{aligned}
$$

Obviously, the reduction is feasible in LOGSPACE. Before showing its correctness, a short sketch of its intuition is given: G_{cols} encodes the three colors, and G_{v1} the vertices contained in V_1. Further, G_{col1} encodes for each $v_i \in V_1$ all possible colorings, and G_{col2} does so for all vertices in $V_2 \cup V_3$. Finally G_{e2} contains all valid edge colorings while G_{e1} stores the invalid ones, and G_{neq} mimics the $\neq$ predicate for the three colors. Now the idea of the reduction can be summarized as follows. Every subgraph $G' \subset G$ that satisfies $\mathcal{C}$, and for which $Cl_{\mathcal{R}}(G) = Cl_{\mathcal{R}}(G')$ holds, does not contain a triple from G_{e1}, that is the encoding of an invalid edge coloring. Further it encodes a single coloring on V_1 by containing exactly one triple from G_{col1} for every $v_i \in V$, and finally $G_{cols} \cup G_{v1} \cup G_{col2} \cup G_{e2} \cup G_{neq} \subset G'$. This is achieved on the one hand by the set of rules $\mathcal{R}$ that only allow to derive triples from G_{col1} and G_{e1}, and on the other hand by the constraints in $\mathcal{C}_0$ and $\mathcal{C}_{col1}$ that require $G' = G$ if $G_{e1} \cap G' \neq \emptyset$ or G' contains more than one triple from G_{col1} for a single $v_i \in V_1$. Finally, $\mathcal{C}_{\mathcal{G}}$ is satisfied over G' exactly if the coloring σ_1 on V_1 encoded by G' is such that for every coloring σ_2 on V_2 there exists a coloring σ_3 on V_3 such that $\sigma_1 \cup \sigma_2 \cup \sigma_3$ is a valid 3-coloring of (V, E). To see this, just note that there exists exactly one way to map G_1 into G' while every possible coloring on V_2 is reflected by a mapping $G_2 \rightarrow G'$. Further, a mapping from G_3 into G' encodes a 3-coloring on V_3. Now G_4 can be mapped into G' if the assignment on $C_1, \ldots, C_n$ reflects a valid 3-coloring for (V, E).

To conclude, the alternating quantifiers in the instance of the Q-3COL$_{\exists,3}$ problem are "encoded" in the above instance of MINI-RDF$^{\subseteq}$ as follows: the existential quantification over colorings of V_1 is encoded by the selection of the proper subgraph G' of G. The universal quantification over colorings of V_2 corresponds to the evaluation of the tgd $\mathcal{C}_{\mathcal{G}}$; indeed, checking if a tgd holds requires to inspect *all* possible homomorphisms from the antecedent to the graph G'. Finally, also the existential quantification over colorings of V_3 is contained in the evaluation of the tgd $\mathcal{C}_{\mathcal{G}}$; indeed, for each homomorphism of the antecedent into the RDF graph G' it must be checked if *there exists* a homomorphism from the conclusion into G'. Note that the colorings of V_2 (referred to by G_2) "occur" in the antecedent of $\mathcal{C}_{\mathcal{G}}$ while the colorings of V_3 (referred to by G_3) "occur" in the conclusion. Actually, also the colorings of V_1 (referred to by G_1) occur in the antecedent. However, here no universal quantification takes place since G' only contains one possible color (i.e., one triple with subject v_i and predicate a) for each vertex in V_1.

The proof of the correctness of the reduction is given in the appendix. □

3.2.2. NP-*complete settings*

As already pointed out above, the main reason why in settings without arbitrary tgds the complexity drops by two levels in the polynomial hierarchy is that testing if some subgraph $G' \subset G$ satisfies $\mathcal{C}$ is now possible in polynomial time, and no longer requires a Π_2^P oracle.

Lemma 3.6. *Both,* MINI-RDF$^{\models}(G, \mathcal{R}, \mathcal{C})$ *and* MINI-RDF$^{\subseteq}(G, \mathcal{R}, \mathcal{C})$*, for arbitrary sets of rules $\mathcal{R}$ and sets $\mathcal{C}$ of b-bounded tgds, are in* NP.

Proof. The problems can be decided by the algorithms depicted in the proof of Lemma 3.4, but (by [19]/Proposition 3) step (2) now requires only polynomial time, hence no call to an oracle is needed. □

Hence it remains to show the NP-hardness for three cases. For the first case MINI-RDF$^{\subseteq}(G, \mathcal{R}, \mathcal{C})$ with fixed sets $\mathcal{R}$ and $\mathcal{C}$, the proof is by reduction from 3COL. Similarly to the Σ_3^P-hardness proof, the idea of this reduction is to follow the steps from the algorithm sketched in the membership proof. "Guessing" a subset of the RDF triples corresponds to defining a coloring on the given graph. On the other hand, the (now tractable) steps of testing if this subset satisfies $\mathcal{C}$ and if it allows one to derive all removed RDF triples via $\mathcal{R}$ correspond to checking if this coloring is indeed a valid 3-coloring. Note that because $\mathcal{C}$ is fixed, it is no longer possible to encode the graph structure into it, but only tests like "if two nodes are connected by an edge, then they must be assigned different colors".

Lemma 3.7. *The problem* MINI-RDF$^{\subseteq}(G,\mathcal{R},\mathcal{C})$ *where both,* $\mathcal{R}$ *and* $\mathcal{C}$ *are considered to be fixed, is* NP*-hard.*

Proof. MINI-RDF$^{\subseteq}(G,\mathcal{R},\mathcal{C})$. For this problem, the NP-hardness is proven by reduction from the problem 3COL. First, consider the following fixed set of rules and constraints, $\mathcal{R}$ and $\mathcal{C}$, respectively.

$$\begin{aligned}
\mathcal{R} = \{&\{X\ a\ C\,.\,D\ iscol\ D\} \Rightarrow \{X\ a\ D\};\\
&\{D\ iscol\ D\} \Rightarrow \{D\ neq\ D\}\}\\
\mathcal{C}\ = \{&\{X\ a\ C\,.\,X\ a\ D\,.\,C\ neq\ D\} \Rightarrow \{0\ neq\ 0\};\\
&\{C\ neq\ C\,.\,X\ a\ D\,.\,F\ iscol\ F\} \Rightarrow \{X\ a\ F\};\\
&\{C\ neq\ C\,.\,F\ iscol\ F\} \Rightarrow \{F\ neq\ F\};\\
&\{X\ e\ Y\,.\,X\ a\ C\,.\,Y\ a\ D\} \Rightarrow \{C\ neq\ D\}\}
\end{aligned}$$

Now let an arbitrary instance of 3COL be given by the graph $\hat{G} = (V, E)$ with $V = \{v_1, \ldots, v_n\}$. Define an instance $(G,\mathcal{R},\mathcal{C})$ of MINI-RDF$^{\subseteq}$ as follows. Let $G = G_{cols} \cup G_v \cup G_e \cup G_{eq} \cup G_{neq}$ where

$$\begin{aligned}
G_{cols} &= \{0\ iscol\ 0\,.\,1\ iscol\ 1\,.\,2\ iscol\ 2\}\\
G_v &= \{v_i\ a\ 0\,.\,v_i\ a\ 1\,.\,v_i\ a\ 2 \mid v_i \in V\},\\
G_e &= \{v_i\ e\ v_j \mid \{v_i, v_j\} \in E\}, \text{ and}\\
G_{eq} &= \{0\ neq\ 0\,.\,1\ neq\ 1\,.\,2\ neq\ 2\},\\
G_{neq} &= \{0\ neq\ 1\,.\,0\ neq\ 2\,.\,2\ neq\ 0\,.\,2\ neq\ 1\,.\\
&\qquad 1\ neq\ 2\,.\,1\ neq\ 0\},
\end{aligned}$$

and a new URI v_i is introduces for every node $v_i \in V$ (by slight abuse of notation, v_i is used to denote both, nodes in V and URIs in G). This reduction is obviously feasible in LOGSPACE. The idea of the reduction is as follows. First of all, the only purpose of G_{eq} is to guarantee that G satisfies $\mathcal{C}$. On the other hand, $G' \cap G_{eq} = \emptyset$ holds for every "valid" (in the sense of MINI-RDF$^{\subseteq}$) subgraph $G' \subset G$. This is achieved by the second and third tgd, that informally state that either $G' \cap G_{eq} = \emptyset$ or $G' = G$. Further, the idea is that G' contains for every $v_i \in V$ exactly one of the three triples $v_i\ a\ 0$, $v_i\ a\ 1$, and $v_i\ a\ 2$: At least one of the three triples must be in G' since otherwise $G_v \not\subseteq Cl_{\mathcal{R}}(G')$, and if G' contains more than one of these triples, then $G' = G$ must hold in order to satisfy the first three tgds. Therefore G' encodes a 3-coloring of V, which is a valid 3-coloring of $\hat{G}$ iff G' satisfies the last tgds.

The proof of the correctness of this reduction can be found in the appendix. □

The remaining two cases for follow immediately from well-known complexity results.

Lemma 3.8. *The problem* MINI-RDF$^{\subseteq}(G,\mathcal{R},\mathcal{C})$*, for arbitrary sets* $\mathcal{R}$ *and no constraints (i.e.* $\mathcal{C} = \emptyset$*) is* NP*-hard.*

Proof. The hardness can be immediately shown by exploiting that testing whether some Datalog rule $r_i\colon G_i \Rightarrow \{t_i\}$ is applicable (i.e. whether there exists a homomorphism $G_i \to G$) is already NP-complete [12]. □

Lemma 3.9. *The problem* MINI-RDF$^{\models}(G,\mathcal{R},\mathcal{C})$ *for a fixed set* $\mathcal{R}$ *and no constraints (i.e.* $\mathcal{C} = \emptyset$*) is* NP*-hard.*

Proof. The NP-hardness, even for $\mathcal{R} = \mathcal{C} = \emptyset$, follows easily from the coNP-hardness of testing if G is lean [12]: $(G, \emptyset, \emptyset)$ is a positive instance of MINI-RDF$^{\models}$ iff there exists some $G' \subset G$ such that $G' \models G$, which is exactly the case if G is not lean. □

3.2.3. Tractable cases

Note that one reason for the settings considered so far being intractable was that because of the constraints it was necessary to in fact check for every possible subgraph of the input G, if it satisfies $\mathcal{C}$ and allows to derive the original information via $\mathcal{R}$. The reason for this is that it might well be the case that some subgraph $G' \subset G$ does not satisfy $\mathcal{C}$, while some subgraph $G'' \subset G'$ satisfies $\mathcal{C}$. This occurs if in G' there exists some homomorphism from the antecedent of a tgd into G' that cannot be extended to the consequent of the tgd. However, by removing further triples from G', for some $G'' \subset G'$ there no longer exists the homomorphism from the antecedent into G'', hence G'' now satisfies this tgd.

However, if there are no constraints at all, such a case cannot occur, and therefore it suffices just to check all subgraphs $G' \subset G$ missing exactly one triple of G. If in addition $\mathcal{R}$ is such that the closure can be computed efficiently, MINI-RDF$^{\subseteq}$ becomes tractable.

Lemma 3.10. *The problem* MINI-RDF$^{\subseteq}(G,\mathcal{R},\mathcal{C})$ *can be decided in* PTIME *if all rules in* $\mathcal{R}$ *are b-bounded and there are no constraints (i.e.* $\mathcal{C} = \emptyset$*).*

Proof. The problem can be decided by testing for every triple $t \in G$, whether it can be removed from G: For every $t \in G$, test whether $t \in Cl_{\mathcal{R}}(G) \setminus \{t\}$. If this is the case for some triple t, then there obviously exists some smaller graph, hence the answer is *Yes*. Otherwise, as no triple in G can be derived via the rules in $\mathcal{R}$ from the remaining triples, the answer is *No*. By [19]/Proposition 9, the check requires only polynomial time. □

4. Rule minimisation

Given the general setting considered so far, another natural question is if the set of rules contains some redundancies. Note that unlike the traditional problems like query containment or several implication problems, the question is not if some rules are redundant (or implied by the other rules) over all possible datasets. Instead the question is posed with a specific dataset in mind. One situation where this may be of interest is to decide if over some dataset indeed all rules must be considered for deriving data. Knowing that some rules can be omitted without losing information may speed up query answering.

Note that rules for RDF, when written as Datalog rules, have a fixed predicate arity of three, which makes reasoning problems computationally easier than in the general Datalog setting (see, e.g. [8]). Although there is a huge amount of literature in the Datalog world addressing related problems (as query containment), the particular nature of the problems studied in this paper requires a distinguished complexity analysis. Depending on whether the Datalog rules are considered as b-bounded or not, the complexity of the problems studied in this section ranges between tractability and $\Delta_2^P[\log n]$-completeness.

The rule minimisation problem is formally defined as follows. Note that since the RDF graph remains unchanged, constraints are irrelevant here.

Definition 4.1. *Let* RDF-RULEMIN$^{\models}(G, \mathcal{R})$ *be the following decision problem:*
INPUT: An RDF graph G and a set $\mathcal{R}$ of RDF rules.
QUESTION: Does there exist $\mathcal{R}' \subset \mathcal{R}$ such that $Cl_{\mathcal{R}'}(G) \models Cl_{\mathcal{R}}(G)$?

Definition 4.2. *Let* RDF-RULEMIN$^{\subseteq}(G, \mathcal{R})$ *be the following decision problem:*
INPUT: An RDF graph G and a set $\mathcal{R}$ of RDF rules.
QUESTION: Does there exist $\mathcal{R}' \subset \mathcal{R}$ such that $Cl_{\mathcal{R}'}(G) = Cl_{\mathcal{R}}(G)$?

4.1. Overview of results

The following theorem summarises the complexity results for the two problems of interest. Thereby two settings are considered, namely $\mathcal{R}$ containing b-bounded or arbitrary rules.

Theorem 4.1. *For* RDF-RULEMIN$^{\models}(G, \mathcal{R})$ *and* RDF-RULEMIN$^{\subseteq}(G, \mathcal{R})$*, the complexity with respect to the rules in $\mathcal{R}$ being b-bounded or not is as depicted in Table 2.*

Before these results are formally proved in Section 4.2, the intuition behind these results is sketched.

First of all, consider the case that all rules in $\mathcal{R}$ are b-bounded. Then, given some RDF graph G, the closure of G under $\mathcal{R}$ can be computed efficiently. In order to check if some rules are redundant, it suffices to check for each $r \in \mathcal{R}$ if it is redundant. That is, it suffices to compare the closure of G under $\mathcal{R}$ with the closure of G under every subset of $\mathcal{R}$ missing exactly one rule. This gives PTIME-membership for RDF-RULEMIN$^{\subseteq}$, since checking if one closure is contained in another one is easy. For RDF-RULEMIN$^{\models}$, comparing two closures means to check if one closure entails the other closure. Since checking entailment is NP-complete, this only gives membership for NP. In fact, it will be shown that the full complexity of testing entailment cannot be avoided, and thus RDF-RULEMIN$^{\models}$ is NP-complete as well.

Allowing for arbitrary rules, note that computing the closure of an RDF graph with respect to a given set of rules is neither in NP nor in coNP: Deciding if certain triples can be derived by the rules is an NP-hard problem. On the other hand, checking if some set of triple patterns is indeed the closure (that is, if no more rule can be applied) is coNP-hard. However, the problem can be easily solved in polynomial time by a deterministic Turing machine having access to an NP-oracle. In fact, a short reflection on the algorithm reveals that the necessary oracle calls are non-adaptive, i.e. completely independent of each other. For RDF-RULEMIN$^{\subseteq}$, this immediately shows that the problem is in $\mathrm{P}_{\parallel}^{\mathrm{NP}}$, hence $\Delta_2^P[\log n]$. RDF-RULEMIN$^{\models}$ on the other hand still requires an entailment test, which obviously is not independent of the closures. However, the problem remains in $\Delta_2^P[\log n]$ nevertheless.

In order to reduce the complexity of the problems RDF-RULEMIN$^{\subseteq}$ and RDF-RULEMIN$^{\models}$, one could seek for approximations of those problems. In fact, one option is to check for *redundant* rules in the set $\mathcal{R}$ of given Datalog rules; or whether some rule is sub-

		RDF-RULEMIN$^{\models}$	RDF-RULEMIN$^{\subseteq}$
(1)	$\mathcal{R}$ bb.	NP-complete	in P
(2)	$\mathcal{R}$ arb.	Δ_2^P $[O(\log(n))]$-complete	Δ_2^P $[O(\log(n))]$-complete

Table 2

The complexity of RDF-RULEMIN$^{\models}$ and RDF-RULEMIN$^{\subseteq}$ depending on whether the rules in $\mathcal{R}$ are b-bounded ("bb") or not ("arb.")

sumed by another rule from $\mathcal{R}$. The first problem is known to be tractable while the test for rule subsumption is NP-complete (see [9]). The latter result can be shown to hold also for rules of bounded arity (which we deal with here), but becomes tractable in the case of b-bounded rules. Further methods (for example, folding and unfolding of rules) are well understood for logic programs (see [25]), and could also apply to the present domain. An in-depth analysis how to use those results in the setting considered in this paper is left for future work.

As mentioned earlier, the rules dealt with in this paper can be seen as Datalog rules using a ternary predicate. Note that the complexity of evaluating Datalog programs which have their (intensional) predicates' arity bounded by a constant has been carefully investigated in [8]. In particular, it is shown there that the general EXPTIME-complexity for reasoning in Datalog programs drops down to complexity classes NP (resp. D^P), thus mirroring the results discussed here and by Meier [19].

4.2. Complexity of rule minimisation

The first lemma formally shows the complexity results in the case of $\mathcal{R}$ containing only b-bounded rules. Thereby the membership is shown using the algorithm sketched before. The hardness proof is again by reduction from 3COL. Its idea is to use a single rule whose application adds an encoding of the edge relation to the RDF data set. Now the RDF triples are defined in such a way that every homomorphism mapping these additional triples into the original set of triples (and therefore witnessing the entailment) defines a valid 3-coloring on the graph. Hence the only rule is redundant iff the graph is 3-colorable.

Lemma 4.2. *For a set $\mathcal{R}$ of b-bounded rules (for fixed b), the problem* RDF-RULEMIN$^{\models}(G,\mathcal{R})$ *is* NP-*complete while* RDF-RULEMIN$^{\subseteq}(G,\mathcal{R})$ *is in* PTIME.

Proof. RDF-RULEMIN$^{\subseteq}(G,\mathcal{R})$. Consider the following algorithm to test whether $\mathcal{R}$ contains redundant rules:

```
1. Compute $G_{clos} = Cl_{\mathcal{R}}(G)$
2. For every $r \in \mathcal{R}$
   (a) Set $\mathcal{R}' = \mathcal{R} \setminus \{r\}$
   (b) Compute $G' = Cl_{\mathcal{R}'}(G)$
   (c) If $G' = G_{clos}$ then return "Yes"
3. Return "No"
```

The algorithm is obviously correct. To see that it also runs in polynomial time, just note that since $\mathcal{R}$ is b-bounded, the closure of G under (a subset of) $\mathcal{R}$ can be computed in polynomial time. Hence, the entire algorithm obviously works in polynomial time.

RDF-RULEMIN$^{\models}(G,\mathcal{R})$. For the *membership*, consider the following changes on the above algorithm. Step (2c) is replaced by the check if $G' \models G_{clos}$ holds. Moreover, in order to avoid multiple entailment checks, the loop in step (2) over all $r \in \mathcal{R}$ is replaced by guessing an appropriate rule $r \in \mathcal{R}$. Hence the witness guessed consists of a rule $r \in \mathcal{R}$ and a homomorphism $G_{clos} \to G'$. This obviously gives a correct NP-algorithm.

The *hardness* is shown by reduction from 3COL. Hence let an arbitrary instance of 3COL be given by the graph (V, E) with $V = \{v_1, \ldots, v_n\}$. Further let d, e be URIs and $\vec{X} = \{X_1, \ldots, X_n\}$ a set of blank nodes. Then the RDF graph G and rule set $\mathcal{R}$ are defined as follows:

$$
\begin{aligned}
G = \{&0\ d\ 1\ .\ 0\ d\ 2\ .\ 1\ d\ 2\ .\ 1\ d\ 0\ .\ 2\ d\ 0\ .\ 2\ d\ 1\ .\\
&0\ e\ 1\ .\ 0\ e\ 2\ .\ 1\ e\ 2\ .\ 1\ e\ 0\ .\ 2\ e\ 0\ .\ 2\ e\ 1\} \cup\\
&\{X_\alpha\ d\ X_\beta \mid (v_\alpha, v_\beta) \in E\}\\
\mathcal{R} = \{&\{Z_1\ d\ Z_2\} \Rightarrow \{Z_1\ e\ Z_2\}\}
\end{aligned}
$$

Clearly, this reduction is feasible in LOGSPACE. For its correctness, observe that $Cl_{\mathcal{R}}(G) = G \cup \{X_\alpha\ e\ X_\beta \mid (v_\alpha, v_\beta) \in E\}$. It remains to show that $G \models Cl_{\mathcal{R}}(G)$ holds (that is r is redundant) iff (V, E) is 3-colorable:

First suppose that $G \models Cl_{\mathcal{R}}(G)$ holds, this means there exists a homomorphism $h\colon Cl_{\mathcal{R}}(G) \to G$. Clearly, this homomorphism must map every triple $X_\alpha\ e\ X_\beta$ to one of the triples $0\ e\ 1\ .\ 0\ e\ 2\ .\ 1\ e\ 2\ .\ 1\ e\ 0$ $.\ 2\ e\ 0\ .\ 2\ e\ 1$. This immediately gives a valid 3-

coloring σ of (V, E) by defining $\sigma(v_i) = h(X_i)$ for every $i \in [n]$.

Conversely, suppose that there exists a valid 3-coloring σ of (V, E). Then consider the mapping h defined on the blank nodes in $\vec{X}$ by setting $h(X_i) = \sigma(v_i)$. It is easy to check that h is indeed a homomorphism from $Cl_{\mathcal{R}}(G)$ to G. $\square$

The following lemma shows the $\Delta_2^P[\log n]$-completeness for RDF-RULEMIN$^{\subseteq}$ in the presence of arbitrary rules. As already indicated before, the idea of the membership proof is to first provide an algorithm for solving the problem that runs in polynomial time on a deterministic Turing machine using polynomially many calls to an NP-oracle. In a second step it is then shown that these calls are in fact non-adaptive. The corresponding hardness proof, which is by reduction from ODD CLIQUE, follows precisely this intuition. Its idea is – given a graph with n nodes – to define for every $i \in \{1 \dots n\}$ a nonredundant rule that creates a unique triple iff the graph contains a clique of size i. Intuitively, evaluating these rules requires the NP-oracle. In addition, there exists another rule that creates this unique triple for each odd size i iff the corresponding triple for $i - 1$ exists. Hence this rule is redundant iff the size of the biggest clique is odd.

Lemma 4.3. *For a set $\mathcal{R}$ of arbitrary rules,* RDF-RULEMIN$^{\subseteq}(G, \mathcal{R})$ *is* $\Delta_2^P[\log n]$*-complete.*

Proof. For the *membership*, consider the following notation: Let G be an arbitrary RDF graph and $\mathcal{R} = \{r_1, \dots, r_n\}$. Then define $\mathcal{R}_i$ for each $i \in [n]$ as $\mathcal{R}_i = \mathcal{R} \setminus \{r_i\}$. Finally recall the definition of the active domain AD. The following (deterministic) algorithm, using an NP-oracle, solves the problem in polynomial time:

```
1. For every $\mathcal{R}' \in \{\mathcal{R}, \mathcal{R}_1, \dots, \mathcal{R}_n\}$
   (a) Set $G_{\mathcal{R}'} = G$
   (b) For every $t \in AD^3$
       – Check (by an NP-oracle) if $t \in Cl_{\mathcal{R}'}(G)$
       – If $t \in Cl_{\mathcal{R}'}(G)$, set $G_{\mathcal{R}'} = G_{\mathcal{R}'} \cup \{t\}$
2. Check if $G_{\mathcal{R}} = G_{\mathcal{R}_i}$ for some $i \in [n]$.
```

To see that this algorithm indeed runs in polynomial time, just note that $Cl_{\mathcal{R}_i}(G) \subseteq Cl_{\mathcal{R}}(G) \subseteq AD \times AD \times AD$. This bounds the total number of iterations of the loop in step (1b) by $(n + 1) \cdot |AD|^3$, hence by $O(pol(n))$. Finally, note that $t \in Cl_{\mathcal{R}'}(G)$ (step (1a)) can be indeed decided in NP: Just consider step (4) from the algorithm presented in the proof of Lemma 3.4. The oracle computes this step and returns "yes" if $t \in \hat{G}_k$, and "no" otherwise. It is easy to see that such an oracle returns "yes" iff $t \in Cl_{\mathcal{R}'}(G)$. Hence the presented algorithm indeed runs in polynomial time. Its correctness follows immediately.

To see that the above algorithm indeed shows $\Delta_2^P[\log n]$-membership, just note that the calls to the oracle are independent of each other. Hence the algorithm uses a polynomial number of non-adaptive NP oracle calls. This proves the membership for $\mathrm{P}_{\|}^{\mathrm{NP}}$, which is equivalent to $\Delta_2^P[\log n]$.

The *hardness* is shown by reduction from the problem ODD CLIQUE. Hence let an arbitrary instance of this problem be given by a graph $\hat{G} = (V, E)$ with $V = \{v_1, \dots, v_n\}$, and assume without loss of generality that $E \neq \emptyset$. Define an instance $(G, \mathcal{R})$ of RDF-RULEMIN$^{\subseteq}(G, \mathcal{R})$ as follows. Let $G = G_{e1} \cup G_{e2} \cup G_{eh} \cup G_c$ where

$$\begin{aligned}
G_{e1} &= \{v_\alpha\ e\ v_\beta \mid (v_i, v_j) \in E, \\
&\qquad\qquad \alpha = \min(i,j), \beta = \max(i,j)\}, \\
G_{e2} &= \{v_i\ e^*\ v_j \mid 1 \leq i < j \leq n\}, \\
G_{eh} &= \{e\ s\ e\ .\ e^*\ s\ e^*\}, \\
G_c &= \{e_i\ c\ o_{i+1} \mid 2 \leq i \leq n \text{ and } i \bmod 2 = 0\},
\end{aligned}$$

and v_i is a new URI for each $v_i \in V$ (by slight abuse of notation, v_i will be used to denote both, vertices in V and URIs in G). Next, for $i \in \{2, \dots, n\}$, consider the following rules. If $i \bmod 2 = 1$, then

$$r_i = \{E\ s\ E\} \cup \{X_r\ E\ X_s \mid 1 \leq r < s \leq i\} \Rightarrow \{E\ o_i\ E\}$$

and if $i \bmod 2 = 0$ then

$$r_i = \{E\ s\ E\} \cup \{X_r\ E\ X_s \mid 1 \leq r < s \leq i\} \Rightarrow \{E\ e_i\ E\}$$

Using these rules, define

$$\begin{aligned}
\mathcal{R} = &\{r_i \mid 1 \leq i \leq n\} \cup \\
&\{\{E\ c\ O\ .\ e\ E\ e\} \Rightarrow \{e\ O\ e\}\}
\end{aligned}$$

The following property, which is shown in the appendix, is crucial for both, the correctness and the understanding of the reduction:

Claim 1: Let $\hat{G}$ be an arbitrary instance of ODD CLIQUE and $(G, \mathcal{R})$ be defined as above. Then none of the rules $r_i \in \mathcal{R}$ for $i \in [n]$ is redundant.

The intuition of the reduction is now as follows: G_{e1} encodes the edge relation E, while G_{e2} encodes a complete graph with n nodes. The idea is that there exists a homomorphism from G_i into G_{e1} if $\hat{G}$ indeed contains a clique of size i. On the other hand, there always exists a homomorphism from G_i into G_{e2} for $i \leq n$. Now denote with m the size of the biggest clique in $\hat{G}$. Then obviously there also exist cliques of all sizes $1 \leq i \leq m$. Hence all rules r_i for $(1 \leq i \leq m)$ can be applied to G. Now if m is odd, then for every triple $e\ e_i\ e$ in $Cl_{\mathcal{R}}(G)$ derived via r_i there also exists a triple $e\ o_{i+1}\ e$ in $Cl_{\mathcal{R}}(G)$ derived via r_{i+1}. Hence the additional rule in $\mathcal{R}$ is redundant. On the other hand, if m is even, then there exists some triple $e\ e_i\ e$ in $Cl_{\mathcal{R}}(G)$ s.t. $e\ o_{i+1}\ e$ cannot be derived by rule r_{i+1}. Hence the only possibility to derive this triple is by the additional rule in $\mathcal{R}$. Therefore no rule in $\mathcal{R}$ is redundant.

The correctness of the reduction is shown in the appendix. □

To conclude the proof of Theorem 4.1, the next lemma shows that the computational complexity remains unchanged for RDF-RULEMIN$^{\models}$ compared to RDF-RULEMIN$^{\subseteq}$. For the hardness it is rather easy to see that the proof from Lemma 4.3 also applies in this case. Extending the membership proof from the previous lemma however no longer leads to only non-adaptive calls to an NP oracle. Nevertheless, the problem can still be shown to be in $\Delta_2^P[\log n]$.

Lemma 4.4. *For a set $\mathcal{R}$ of arbitrary rules,* RDF-RULEMIN$^{\models}(G, \mathcal{R})$ *is $\Delta_2^P[\log n]$-complete.*

Proof. In order to show the *membership*, recall the algorithm presented in the proof of Lemma 4.3. For RDF-RULEMIN$^{\models}$, instead of checking in step (2) if $G_{\mathcal{R}} = G_{\mathcal{R}_i}$ for some $i \in [n]$, it is now necessary to check if $G_{\mathcal{R}_i} \models G_{\mathcal{R}}$ for some $i \in [n]$, which cannot be done in polynomial time. In order to solve the problem it is necessary to replace step (2) by

2. *Check (by an* NP*-oracle) if $G_{\mathcal{R}_i} \models G_{\mathcal{R}}$ for some $i \in [n]$*

In order to decide the problem, the oracle just guesses $G_{\mathcal{R}_i}$ and a homomorphism $h\colon G_{\mathcal{R}} \to G_{\mathcal{R}_i}$. Hence the correctness of this modification follows immediately. Therefore it only remains to show that the problem can still be solved in polynomial time by a deterministic Turing machine using at most $O(\log(|G|+|\mathcal{R}|))$ oracle calls.

This is trivially still true for step (1). Step (2) requires a single oracle call only, but this call depends on the result of the calls in step (1). Hence not all oracle calls are non-adaptive. However, by the equivalence of $\mathrm{P}_{\|}^{\mathrm{NP}}$ and $\Delta_2^P[\log n]$, there exists a deterministic Turing machine on which step (1) runs in polynomial time using $O(\log(|G|+|\mathcal{R}|)$ adaptive oracle calls. It follows immediately that this machine can be modified such that it also performs the check in step (2), and that this modification does not change the bound of $O(\log(|G| + |\mathcal{R}|)$ oracle calls.

For the *hardness*, note that the hardness proof in Lemma 4.3 also shows hardness for MINI-RDF$^{\models}$. Both, the graph G defined by the reduction and $Cl_{\mathcal{R}}(G)$ do not contain any blank node. Hence $Cl_{\mathcal{R}}(G') \models Cl_{\mathcal{R}}(G)$ iff $Cl_{\mathcal{R}}(G) \subseteq Cl_{\mathcal{R}}(G')$, which concludes the proof. □

5. Minimisation in the presence of queries

So far, the aim of the problems was always to minimise the given RDF graph only in such a way that no information is lost. However, in practice, this might be more restrictive than necessary. For example, if data is transferred into some RDF store that can only be accessed through some narrow query interface. In such situations, it might be the case that some information cannot be accessed anyway. Hence it would be save to remove all this information. Obviously, this increases the potential for minimisation.

Therefore the variant of the RDF graph and rule minimisation problems considered in this section guarantees completeness only with respect to a given set of queries. Thereby the focus lies on (unions of) conjunctive queries (CQs resp. UCQs). Formally, the following two problems are studied:

Definition 5.1. MINI-RDF$^{\subseteq,CQ}(G, \mathcal{R}, \mathcal{C}, \mathcal{Q})$ *is the following decision problem:*
INPUT: An RDF graph G, a set $\mathcal{R}$ of RDF rules, a set $\mathcal{C}$ of tgds such that G satisfies $\mathcal{C}$, and a set $\mathcal{Q}$ of CQs.
QUESTION: Is there a $G' \subset G$ such that (1) for every $q \in \mathcal{Q}$, the answers to q over $Cl_{\mathcal{R}}(G)$ coincide with the answers to q over $Cl_{\mathcal{R}}(G')$ and (2) G' satisfies $\mathcal{C}$?

Definition 5.2. RDF-RULEMIN$^{\subseteq,CQ}(G, \mathcal{R}, \mathcal{Q})$ *is the following decision problem:*
INPUT: An RDF graph G, a set $\mathcal{R}$ of RDF rules, and a set $\mathcal{Q}$ of CQs.
QUESTION: Is there a $\mathcal{R}' \subset \mathcal{R}$ s.t. for every $q \in \mathcal{Q}$, the answers to q over $Cl_{\mathcal{R}}(G)$ coincide with the answers to q over $Cl_{\mathcal{R}'}(G)$?

Note that the result of a CQ not necessarily represents a valid RDF graph – for instance, take a CQ with answer predicate of arity bigger than 3. Hence the notion of RDF-entailment is not applicable to the answers of a CQ. This is the reason why only the extended variants of MINI-RDF$^{\subseteq}$ and RDF-RULEMIN$^{\subseteq}$ are studied here, but not of MINI-RDF$^{\models}$ and RDF-RULEMIN$^{\models}$.

As already pointed out, there is hope that maintaining only those information actually contributing to the result of the queries allows to further reduce the size of the stored RDF graph. However, it turns out that for most of the settings considered in the previous sections, this additional optimisation potential does not come for free. In many cases, the computational complexity of determining if some RDF graph can be further reduced or not increases compared to the problems in Section 3 and Section 4. To be able to better understand how queries influence the computational complexity of the problem, queries are considered to be either body-b-bounded, head-b-bounded, or arbitrary (see Section 2 for the definition of these concepts). However, it turns out that in some cases already body-b-bounded queries are enough to increase the complexity of the problem.

5.1. Overview of results

A simple observation reveals that the problems RDF-RULEMIN$^{\subseteq}$ and MINI-RDF$^{\subseteq}$ are special cases of the problems RDF-RULEMIN$^{\subseteq,CQ}$ and MINI-RDF$^{\subseteq,CQ}$, respectively. Just consider the simple query $\{S\ P\ O\} \to ans(S,P,O)$. Then, given some RDF graph G, in order to return over some subgraph $G' \subset G$ all answers from q over G it must be possible to derive G from G'. Hence, all hardness results from the previous sections carry over. Thus adding the CQ $\mathcal{Q}$ to the input of the problem does not make the problem easier.

In addition to the various settings studied in the previous sections resulting from different restrictions on $\mathcal{C}$ and $\mathcal{R}$, three different settings of each CQ-variant of these problems are studied by considering $\mathcal{Q}$ to be body-b-bounded, head-b-bounded, or unrestricted, respectively. This gives the following complexity results.

Theorem 5.1. *For* MINI-RDF$^{\subseteq,CQ}$*, the complexity w.r.t. different assumptions on the input (arbitrary, b-bounded or fixed rule set; arbitrary, b-bounded, fixed, or no constraints; body-b-bounded, head b-bounded, or arbitrary CQs) is as depicted in Table 3, rows (1) – (12). Likewise, the complexity of* RDF-RULEMIN$^{\subseteq,CQ}$ *is depicted in Table 3, rows (I) – (II).*

All lower bounds hold even if $\mathcal{Q}$ consists of a single CQ. Likewise, all upper bounds hold even if $\mathcal{Q}$ is a set of UCQs.

In order to compare the new results with those presented in the previous sections, the last column in Table 3 recalls the corresponding results for MINI-RDF$^{\subseteq}$ and RDF-RULEMIN$^{\subseteq}$. Note that for body-b-bounded queries the complexity changes slightly only in the cases where arbitrary rules are allowed, while for head-b-bounded queries no case is below $\Delta_2^P[\log n]$. Hence in this setting, also the previously tractable cases become hard. Finally, in case of arbitrary queries, the complexity heavily increases in most of the settings.

Similar to the case in Section 3, it is not necessary to show each of the results separately. Instead, the proof plan for Theorem 5.1 is composed as follows: Obviously, body-b-bounded (U)CQs are a special case of head-b-bounded (U)CQs, which in turn are a special case of arbitrary (U)CQs. By combining this observation with Lemma 3.2, to prove Theorem 5.1, it suffices to show membership for the entries (8a) (covering the two tractable cases), (6a) (covering all cases in NP), (2b) (covering all cases in $\Delta_2^P[\log n]$), (1c) (covering the Π_2^P-complete cases), (4c) (covering all cases in Σ_3^P) as well as (Ic) (for the settings of RDF-RULEMIN$^{\subseteq,CQ}$ in Π_2^P), (I.b) (for the settings in $\Delta_2^P[\log n]$), and (II.b) (for the tractable case of RDF-RULEMIN$^{\subseteq,CQ}$) in Table 3. On the other hand, for the hardness results it suffices to show hardness for (9a) (covering all cases hard for Σ_3^P where $\mathcal{Q}$ is body-b-bounded or head-b-bounded, (11c) (covering all Σ_3^P-hard cases with arbitrary $\mathcal{Q}$), (12c) (covering all Π_2^P-complete cases), (12b) (covering all $\Delta_2^P[\log n]$-complete cases for head-b-bounded $\mathcal{Q}$), (4a) (covering all $\Delta_2^P[\log n]$-complete cases for body-b-bounded $\mathcal{Q}$), (11a) (covering all NP-complete settings), as well as (II.c) (covering also (Ic)), (I.a) (covering also (Ib)), and (II.b) (covering (Ib) as well).

The proofs of these results are given in Section 5.2 for the graph minimisation problem, and Section 5.3 for the rule minimisation problem. Before, in the remainder of this subsection, an intuitive explanation of the above results is given.

Starting with the membership results, note that for the most general setting (1c), the algorithm used to solve MINI-RDF$^{\subseteq}$ can be adapted to compare the results of the queries instead of the closures of the graphs, without increasing its complexity: It suffices to guess a subset $G' \subset G$ and check with Π_2^P-oracles if G' satisfies $\mathcal{C}$ and if $q(\hat{G}) = q(\hat{G}')$, where $\hat{G} = Cl_{\mathcal{R}}(G)$, and $\hat{G}' = Cl_{\mathcal{R}}(G')$ (note that since $Cl_{\mathcal{R}}(G) \subseteq AD^3$, its size is polynomially bounded in the input size).

		$\mathcal{Q}$ body-bb (a)	$\mathcal{Q}$ head-bb (b)	$\mathcal{Q}$ arb. (c)	MINI-RDF$^{\subseteq}$
(1)	$\mathcal{R}$ arb., $\mathcal{C}$ arb.	Σ_3^P-complete	Σ_3^P-complete	Σ_3^P-complete	Σ_3^P-*complete*
(2)	$\mathcal{R}$ arb., $\mathcal{C}$ bb	$\Delta_2^P[\log n]$-complete	$\Delta_2^P[\log n]$-complete	Σ_3^P-complete	NP-*complete*
(3)	$\mathcal{R}$ arb., $\mathcal{C}$ fixed	$\Delta_2^P[\log n]$-complete	$\Delta_2^P[\log n]$-complete	Σ_3^P-complete	NP-*complete*
(4)	$\mathcal{R}$ arb., $\mathcal{C} = \emptyset$	$\Delta_2^P[\log n]$-complete	$\Delta_2^P[\log n]$-complete	Π_2^P-complete	NP-*complete*
(5)	$\mathcal{R}$ bb., $\mathcal{C}$ arb.	Σ_3^P-complete	Σ_3^P-complete	Σ_3^P-complete	Σ_3^P-*complete*
(6)	$\mathcal{R}$ bb, $\mathcal{C}$ bb	NP-complete	$\Delta_2^P[\log n]$-complete	Σ_3^P-complete	NP-*complete [19]*
(7)	$\mathcal{R}$ bb, $\mathcal{C}$ fixed	NP-complete	$\Delta_2^P[\log n]$-complete	Σ_3^P-complete	NP-*complete*
(8)	$\mathcal{R}$ bb, $\mathcal{C} = \emptyset$	in P	$\Delta_2^P[\log n]$-complete	Π_2^P-complete	*in* P
(9)	$\mathcal{R}$ fixed, $\mathcal{C}$ arb.	Σ_3^P-complete	Σ_3^P-complete	Σ_3^P-complete	Σ_3^P-*complete*
(10)	$\mathcal{R}$ fixed, $\mathcal{C}$ bb	NP-complete	$\Delta_2^P[\log n]$-complete	Σ_3^P-complete	NP-*complete*
(11)	$\mathcal{R}$ fixed, $\mathcal{C}$ fixed	NP-complete	$\Delta_2^P[\log n]$-complete	Σ_3^P-complete	NP-*complete*
(12)	$\mathcal{R}$ fixed, $\mathcal{C} = \emptyset$	in P	$\Delta_2^P[\log n]$-complete	Π_2^P-complete	*in* P
(I.)	$\mathcal{R}$ arb.	$\Delta_2^P[\log n]$-complete	$\Delta_2^P[\log n]$-complete	Π_2^P-complete	$\Delta_2^P[\log n]$-*complete*
(II.)	$\mathcal{R}$ bb.	in P	$\Delta_2^P[\log n]$-complete	Π_2^P-complete	*in* P

Table 3

The complexity of MINI-RDF$^{\subseteq, CQ}$ (1-12) and RDF-RULEMIN$^{\subseteq, CQ}$ (I. - II.) w.r.t. input parameters ("bb" stands for "b-bounded", and "arb." for "arbitrary"). The last column contains the results for MINI-RDF$^{\subseteq}$ and RDF-RULEMIN$^{\subseteq}$ from the previous sections for comparison.

The other columns contain potentially easier settings because of the restrictions on the queries. In particular, when restricting $\mathcal{Q}$ to head-b-bounded CQs, then there are at most polynomially many candidates for answer-tuples for each query. Hence computing the answers to a given query is now feasible in $\Delta_2^P[\log n]$ (note that it is still necessary to test an exponential number of homomorphisms to decide for each candidate if it is indeed a solution). When $\mathcal{Q}$ is restricted to body-b-bounded queries, all answers to a query over a given RDF dataset can be computed efficiently.

On the other hand, the other rows contain potentially easier settings because of restrictions on $\mathcal{R}$ and $\mathcal{C}$. In particular, already restricting $\mathcal{R}$ to b-bounded rules allows to compute the closure of a given graph in polynomial time. Similarly, restricting $\mathcal{C}$ to b-bounded tgds allow to test efficiently if $\mathcal{C}$ is satisfied by some RDF graph. Further, if no constraints are present at all, it suffices to check the "direct" subsets $G' = G \setminus \{t\}$ for each $t \in G$. Thus the non-deterministic guess of $G' \subset G$ is no longer needed. By the same token, rule minimisation is not harder than Π_2^P, since it suffices to check the direct subsets $\mathcal{R}' = \mathcal{R} \setminus \{r\}$.

Turning to the lower bounds, the NP-hardness for (11a) follows immediately from the above remark that the hardness results of MINI-RDF$^{\subseteq}$ carry over. Concerning the $\Delta_2^P[\log n]$-complete settings, note that in order to solve MINI-RDF$^{\subseteq, CQ}$, one must make sure to take indeed all answers to a query into account, and that these answers are indeed computed over the closure of the current RDF dataset (compared to MINI-RDF$^{\subseteq}$, where it was sufficient to show $G \subseteq \hat{G}'$ where $\hat{G}' \subseteq Cl_{\mathcal{R}}(G')$, but not necessarily $\hat{G}' = Cl_{\mathcal{R}}(G')$). This introduces two different reasons for the $\Delta_2^P[\log n]$-hardness. For arbitrary rules $\mathcal{R}$ the problem is that testing if some triple is contained in the closure is NP-hard, while testing if the closure does not contain any more triples is coNP-hard (4a). Similarly, for head-b-bounded queries it is NP-hard to decide if a tuple belongs to the answers of a query, while it is coNP-hard to decide if the set of answers is indeed complete (12b). Combined with the other parameters of the problem, one of these two reasons suffices for MINI-RDF$^{\subseteq, CQ}$ to become $\Delta_2^P[\log n]$-hard.

The key observation for the hardness results for (12c) and (11c) is that in these settings $\mathcal{Q}$ is capable of defining quite powerful constraints on "valid" subgraphs of G. Seen from the perspective of the hardness proof, this allows to "compensate" some of the restrictions considered on $\mathcal{C}$, such that (11c) still contains the full Σ_3^P-hardness. An important aspect of using $\mathcal{Q}$ to express constraints that cannot be defined by b-bounded $\mathcal{C}$ is the possibility to use projection. However, one big difference between $\mathcal{C}$ and $\mathcal{Q}$ seen as constraints is that $\mathcal{Q}$ can only express monotone constraints: Whenever some subgraph $G' \subset G$ does not return a certain answer, then it is ensured that also no subgraph of G' returns this answers. This is the reason why (12c) becomes easier to solve.

For the rule minimisation, the reasons for the hardness results are very similar to those for the corresponding results of the graph minimisation problem.

Note that all of the membership results still hold when allowing UCQs instead of CQs, and some of them even hold for UCQs$^\neg$. Whenever this is easy to see, the corresponding result is stated. However, a detailed analysis of how the complexity is influenced by replacing (U)CQs by (U)CQs$^\neg$ is left to future work.

5.2. Graph minimisation

This section in detail treats the results for the graph minimisation problem with respect to a given set of queries. For convenience, the observation that the hardness results from Section 3 carry over, which was already mentioned before, is stated formally.

Proposition 5.2. *Let $(G, \mathcal{R}, \mathcal{C})$ be an arbitrary instance of* MINI-RDF$^\subseteq$. *Further consider the set $\mathcal{Q} = \{ \{S \ P \ O\} \rightarrow ans(S, P, O)\}$ containing a single query. Then $(G, \mathcal{R}, \mathcal{C})$ is a positive instance of* MINI-RDF$^\subseteq$, *iff $(G, \mathcal{R}, \mathcal{C}, \mathcal{Q})$ is a positive instance of* MINI-RDF$^{\subseteq, CQ}$.

5.2.1. Settings complete for Σ_3^P

The first result shows that for the those settings of MINI-RDF$^\subseteq$ considered in Section 3 that are already Σ_3^P-complete, the additional optimisation potential introduced by MINI-RDF$^{\subseteq, CQ}$ does not further increase the (already high) complexity of the problem. Even when allowing for arbitrary conjunctive queries in addition to arbitrary rules and constraints, the problem can still be solved in Σ_3^P. The reason for this is that on the one hand, deciding if some answers are lost over some smaller graph is not harder than checking if this smaller graph still satisfies all constraints. On the other hand, these two problems do not introduce "orthogonal" sources of complexity, but can be decided independently of each other.

Lemma 5.3. MINI-RDF$^{\subseteq, CQ}(G, \mathcal{R}, \mathcal{C}, \mathcal{Q})$, *for arbitrary sets $\mathcal{R}$ and $\mathcal{C}$, where $\mathcal{Q}$ may contain arbitrary CQs can be solved in Σ_3^P. The problem remains in Σ_3^P even if $\mathcal{Q}$ is a set of arbitrary UCQs$^\neg$.*

Proof. The problem can be decided by the following algorithm using a Π_2^P-oracle:

1. *Compute $\hat{G} = Cl_{\mathcal{R}}(G)$*
2. *Guess a $G' \subset G$*
3. *Check if G' satisfies $\mathcal{C}$ (otherwise return* "no"*)*
4. *Compute $\hat{G}' = Cl_{\mathcal{R}}(G')$*
5. *For every $q \in \mathcal{Q}$, check by a call to the oracle whether for every homomorphism $\tau\colon body(q) \rightarrow \hat{G}$ there exists an extension of τ^h to τ' (where τ^h is τ restricted to $head(q)$ and $\tau'\colon body(q) \rightarrow \hat{G}'$) (i.e. whether every result in $q(\hat{G})$ can be also derived over $\hat{G}'$)*

The correctness of this algorithm follows immediately. To see that this algorithm indeed runs in (nondeterministic) polynomial time, note the following: Step (1) and step (4) fit into $\Delta_2^P[\log n]$ (cf. proof of Lemma 4.3). Step (3) can be decided by a Π_2^P-oracle ([10]), and also step (5) can be obviously decided by a Π_2^P-oracle.

It remains to show that the result still holds even if $\mathcal{Q}$ is a set of UCQs$^\neg$. Towards this goal, first of all note that the above algorithm can be extended to also work for UCQs by replacing step (5) with

5. *For every $Q \in \mathcal{Q}$, check by a call to the oracle, whether for every $q \in Q$ and every homomorphism $\tau\colon body(q) \rightarrow \hat{G}$ there exists a $q' \in Q$ and an extension τ' of τ^h (where τ^h is defined as before) s.t. $\tau'\colon body(q') \rightarrow \hat{G}'$, (i.e. whether every result in $q(\hat{G})$ can be also derived over $\hat{G}'$)*

Obviously this step can still be decided by a Π_2^P-oracle: It suffices to just guess $q \in Q$ together with the homomorphism.

The extension to UCQs$^\neg$ requires two additional steps. On the one hand, to decide step (5), for every guessed homomorphism τ (resp. τ'), it is now necessary to check if $\tau(pos(q)) \subseteq \hat{G}$ (resp. $\tau'(pos(q')) \subseteq \hat{G}'$) and $\tau(neg(q)) \cap \hat{G} = \emptyset$ (resp. $\tau'(neg(q')) \cap \hat{G}' = \emptyset$). Finally, in an additional step, it must be checked (analogously to step (5)) if all results in $q(\hat{G}')$ can also be derived over $\hat{G}$. □

It remains to show Σ_3^P-hardness for two settings. The first one follows immediately from Lemma 3.5 and Proposition 5.2.

Lemma 5.4. MINI-RDF$^{\subseteq, CQ}(G, \mathcal{R}, \mathcal{C}, \mathcal{Q})$, *when $\mathcal{R}$ is fixed, $\mathcal{Q}$ contains only body-b-bounded CQs but $\mathcal{C}$ may contain arbitrary tgds is Σ_3^P-hard, already if $\mathcal{Q}$ contains a single CQ only.*

The second case – (11c) in Table 3 – requires a more involved proof, done by reduction from Q-3COL$_{\exists,3}$. Informally, for an instance $\mathcal{G} = ((V, E), (V_1, V_2, V_3))$ of Q-3COL$_{\exists,3}$, the reduction defines an instance of MINI-RDF$^{\subseteq, CQ}(G, \mathcal{R}, \mathcal{C}, \mathcal{Q})$ where $\mathcal{Q}$ contains a single CQ q that returns over G an encoding of all possible 3-colorings of the vertices in V_2. Further, the rules, constraints, and q are defined in such a way that "guessing a subset" of G has two effects. On the one hand, it fixes a coloring on V_1. On the other hand, q only returns encodings of those colorings on V_2 that can be extended to V_3 in such a way that the combination of the colorings on V_1, V_2, and V_3 gives a valid 3-coloring of V. Hence over every such subset of G, the CQ q returns all possible colorings on V_2 (and therefore the same answers as over G) iff there exists some coloring on V_1 (encoded by the selected subgraph of G) such that for all colorings on V_2 there exists a coloring on V_3 (encoded by the fact that q still returns the same answers) such that the combination of these colorings gives a valid 3-coloring of V.

Lemma 5.5. MINI-RDF$^{\subseteq,CQ}(G,\mathcal{R},\mathcal{C},\mathcal{Q})$*, when both, $\mathcal{R}$ and $\mathcal{C}$ are fixed but $\mathcal{Q}$ may contain arbitrary CQs is Σ_3^P-hard, already if $\mathcal{Q}$ contains a single CQ only. It even remains Σ_3^P-hard for the case where $\mathcal{R} = \emptyset$.*

Proof. The proof is by reduction from Q-3COL$_{\exists,3}$. Hence let $\mathcal{G} = ((V,E),(V_1,V_2,V_3))$ be an arbitrary instance of Q-3COL$_{\exists,3}$ with $V = \{v_1,\dots,v_n\}$ and $|V_2| = p$. Define an instance $(G,\mathcal{R},\mathcal{C},\mathcal{Q})$ of MINI-RDF$^{\subseteq,CQ}(G,\mathcal{R},\mathcal{C},\mathcal{Q})$ as follows. Let $G = G_{cols} \cup G_{v1} \cup C_{c1} \cup G_{e1} \cup G_{e2} \cup G_{neq}$ where

$$\begin{aligned}
G_{cols} &= \{0\ iscol\ 0\,.\,1\ iscol\ 1\,.\,2\ iscol\ 2\}\\
G_{v1} &= \{v_i\ v\ a \mid v_i \in V_1\},\\
G_{c1} &= \{v_i\ c\ 0\,.\,v_i\ c\ 1\,.\,v_i\ c\ 2 \mid v_i \in V_1\},\\
G_{e1} &= \{0\ e\ 0\,.\,1\ e\ 1\,.\,2\ e\ 2\},\\
G_{e2} &= \{0\ e\ 2\,.\,0\ e\ 1\,.\,1\ e\ 2\,.\,1\ e\ 0\,.\,2\ e\ 0\,.\,2\ e\ 1\},\\
G_{neq} &= \{0\ neq\ 2\,.\,0\ neq\ 1\,.\,2\ neq\ 0\,.\,2\ neq\ 1\,.\\
&\qquad 1\ neq\ 2\,.\,1\ neq\ 0\},
\end{aligned}$$

and a new URI v_i is introduced for every $v_i \in V$ (by slight abuse of notation, let v_i denote both, nodes in V and URIs G). As claimed in the lemma, $\mathcal{R} = \emptyset$, and $\mathcal{C}$ contains three fixed tgds

$$\begin{aligned}
\mathcal{C} = \{\ &\{X\ e\ X\,.\,C\ iscol\ C\} \Rightarrow \{C\ e\ C\};\\
&\{X\ e\ X\,.\,Y\ v\ a\,.\,Z\ iscol\ Z\} \Rightarrow \{Y\ c\ Z\}\\
&\{X\ v\ a\,.\,X\ c\ C_1\,.\,X\ c\ C_2\,.\,C_1\ neq\ C_2\}\\
&\qquad\qquad \Rightarrow \{0\ e\ 0\}\}.
\end{aligned}$$

Finally, define Q containing a single query q as

$$\begin{aligned}
\mathcal{Q} = \{\ &\{v_i\ c\ C_i \mid v_i \in V_1\} \cup\\
&\{C_i\ iscol\ C_i \mid v_i \in V_2 \cup V_3\} \cup\\
&\{C_\alpha\ e\ C_\beta \mid (v_\alpha, v_\beta) \in E\} \cup\\
&\{X\ v\ a\,.\,C\ iscol\ C\,.\,D_1\ neq\ D_2\}\\
&\qquad \rightarrow ans(C,X,D_1,D_2,C_{i_1},\dots,C_{i_p})\}
\end{aligned}$$

where $\{C_1,\dots,C_n\}$ contains a new blank node C_i for each vertex $v_i \in V$ and $\{C_{i_1},\dots,C_{i_p}\} \subseteq \{C_1,\dots,C_n\}$ are those variables $C_i \in \{C_1,\dots,C_n\}$ such that $v_i \in V_2$.

The reduction is obviously feasible in LOGSPACE. Before showing its correctness, first its intuition is described. The following three claims will prove useful for both, sketching the intuition of the reduction as well as proving its correctness.

Claim 1: Let $\mathcal{G}$, G, and q as defined above. Consider the following sets: $C_1 = \{0,1,2\}$, $X = \{v_i \mid v_i \in V_1\}$, $D = \{(D_1,D_2) \mid D_1,D_2 \in \{0,1,2\}, D_1 \neq D_2\}$, and $C_2 = \{0,1,2\}^p$. Then $q(G) = C_1 \times X \times D \times C_2$.

Since the above claim follows immediately from the definition of G and q the proof is omitted.

Claim 2: Consider G and $\mathcal{C}$ as defined by the reduction, and for $v_i \in V$ let $G_v^i = \{v_i\ c\ 0\,.\,v_i\ c\ 0\,.\,v_i\ c\ 0\}$. Then for every $G' \subset G$ that satisfies $\mathcal{C}$, the following two properties hold:

1. $G_{e1} \cap G' = \emptyset$,
2. $|G_v^i \cap G'| \leq 1$ for every $v_i \in V$

Proof of Claim 2: The proof is analogous to the proofs of properties (iii) and (iv) in the proof of Lemma 3.7.

Claim 3: Consider G and $\mathcal{C}$ as defined by the reduction, and let G_v^i be as defined in Claim 2. Then for every $G' \subset G$ that satisfies $q(G) = q(G')$, the following properties hold:

1. $G_{cols} \cup G_{v1} \cup G_{neq} \subseteq G'$
2. $|G_v^i \cap G'| \geq 1$ for every $v_i \in V_1$

Proof of Claim 3: Both properties follow immediately from Claim 1 and the assumption that $q(G) = q(G')$.

The intuition of this reduction can be sketched as follows: $|G_v^i \cap G'| = 1$ holds for every subgraph $G' \subset G$ that satisfies $\mathcal{C}$ and such that $q(G) = q(G')$. Then the triples $G_{c1} \cap G'$ encode the required coloring on V_1. The idea is that every mapping $\mu\colon \{C_1,\dots,C_n\}$ that can be extended to μ' on C, X, D_1, D_2 such that μ' is a homomorphism $body(q) \rightarrow G'$ encodes a valid 3-coloring on V (since $G_{e1} \cap G' = \emptyset$, every triple $C_\alpha\ e\ C_\beta$ representing an edge in E must be mapped to a valid edge coloring in G_{e2}). Now G' leaves only a single possible mapping on the C_i's encoding V_1 and every possible mapping on the C_i's representing V_2 must give a valid solution. Hence G' encodes a coloring on V_1 s.t. for every coloring of V_2 there exists a coloring on V_3 such that these three colorings are a valid 3-coloring of (V, E).

To conclude, the quantifier alternation of Q-3COL$_{\exists,3}$ is "encoded" via the single CQ as follows. Omitting C, X, D_1, D_2 for the moment, the requirement that $\{0,1,2\}^p \subseteq q(G')$ can be read as *for all* mappings $\mu\colon \{C_{i_1}, \ldots, C_{i_p}\} \rightarrow \{0,1,2\}$ *there exists* an extension $\mu'\colon \{C_i \mid v_i \in V\} \rightarrow \{0,1,2\}$ of μ that is a homomorphism from $body(q)$ into G' (which corresponds to "for all colorings on V_2, there exists a coloring on $V_1 \cup V_3$"). The correct alternation is expressed by the selection of G': This selection provides only a single possibility for μ' to map $\{C_i \mid v_i \in V_1\}$ into $\{0,1,2\}$. Hence the requirement of q to return $\{0,1,2\}^p$ over G' can be read as "*there exists* a single mapping on $\{C_i \mid v_i \in V_1\}$ such that *for all* mappings on $\{C_i \mid v_i \in V_2\}$ *there exists* a mapping on $\{C_i \mid v_i \in V_3\}$.

The proof of the correctness of this reduction is given in the appendix. □

As already discussed in the previous subsection, queries in $\mathcal{Q}$ can be seen as additional constraints that must be satisfied by subgraphs of G' of the input graph G, if they were already satisfied over G (note that unlike the constraints in $\mathcal{C}$, the "constraints" expressed by $\mathcal{Q}$ need not be satisfied over G, i.e. the queries may return empty results). Intuitively, this shows that the introduction of $\mathcal{Q}$ in combination with b-bounded constraints adds at least some of the expressibility that is lost when restricting $\mathcal{C}$ to contain b-bounded tgds only. The results in the next section show that this combination of $\mathcal{C}$ and $\mathcal{Q}$ is indeed necessary to obtain the full hardness. Note that in the above reduction, $\mathcal{Q}$ is used to express "positive" constraints, that is $\mathcal{Q}$ defines which triples from G must still be contained in every reduced subgraph. On the other hand, the tgds in $\mathcal{C}$ were used to express "negative" constraints, that is which triples from G must not be contained in valid subgraphs $G' \subset G$. To be a little bit more precise, tgds in $\mathcal{C}$ allow to state constraints like "either all triples of a certain kind are in G', or none of them". This is possible, since tgds may be violated by some RDF graph, but satisfied by some proper RDF subgraph (that is either if all triples of a certain set are in G or none). However, this is not the case with queries in $\mathcal{Q}$. If a query does not return a certain answer over some RDF graph, then this answer is also not returned over every (proper) subgraph. This is going to be the key observation for the lower complexity discussed next.

5.2.2. *Settings complete for* Π_2^P

The reason for the settings with $\mathcal{C} = \emptyset$ being easier than those with explicit constraints is the same that was already observed in Section 3 for the cases with b-bounded $\mathcal{R}$ and $\mathcal{C} = \emptyset$ being tractable. Instead of having to test every subset of a given RDF graph, it suffices to just test every subset missing exactly one triple.

Lemma 5.6. MINI-RDF$^{\subseteq,CQ}(G, \mathcal{R}, \mathcal{C}, \mathcal{Q})$, *for arbitrary* $\mathcal{R}$, $\mathcal{C} = \emptyset$ *and where* $\mathcal{Q}$ *may contain arbitrary CQs can be solved in* Π_2^P. *The problem remains in* Π_2^P *even if* $\mathcal{Q}$ *is a set of arbitrary UCQs*$^{\neg}$.

Proof. The lemma is proven by devising a nondeterministic algorithm that decides the co-problem (that is, whether G is already minimal with respect to $\mathcal{R}, \mathcal{C}$ and $\mathcal{Q}$), using a coNP-oracle, in polynomial time. In the following, let $G = \{t_1, \ldots, t_n\}$, and for each $i \in [n]$, let $G_i = G \setminus \{t_i\}$.

1. *Compute* $\hat{G} = Cl_{\mathcal{R}}(G)$
2. *Compute* $\hat{G}_i = Cl_{\mathcal{R}}(G_i)$ *for every* $i \in [n]$
3. *Guess* n *(not necessarily distinct) queries* $q_i \in \mathcal{Q}$
4. *Guess* n *homomorphisms* $\tau_1, \ldots, \tau_n$ *with* $\tau_i\colon body(q_i) \rightarrow \hat{G}$
5. *Check for all* $i \in [n]$ *by a* coNP*-oracle (i.e. using* n *calls):* $\tau_i(head(q_i)) \notin q_i(\hat{G}_i)$.
   I.e. check that for all homomorphisms $\tau_i'\colon body(q_i) \rightarrow \hat{G}_i$ *with* $\tau_i'(body(q_i)) \subseteq \hat{G}_i$, *also* $\tau_i'(head(q_i)) \neq \tau(head(q_i))$
6. *G is minimal, iff all* n *oracle calls return* "yes"

The intuition of this is as follows. After computing the closure of G and all "candidates" G_i, guess for every G_i some query q_i and an answer over $\hat{G}$ to this query, such that the answer cannot be created by q_i over $\hat{G}_i$. If this is the case, G cannot be further minimised.

The correctness of this nondeterministic algorithm follows immediately. Its polynomial runtime follows from the facts that $\hat{G}$ and $\hat{G}_i$ can be computed in $\Delta_2^P[\log n]$, and that the size of $\hat{G}$ (thus also of each $\hat{G}_i$) is at most polynomial in the size of the input (cf. proof of Lemma 4.3).

Concerning an extension of this algorithm to UCQs, it suffices to replace step (3) by

3. *Guess* $Q_i \in \mathcal{Q}$ *and* $q_i \in Q_i$

and in step (5) to check that $\tau_i(head(q_i)) \notin q_i(\hat{G}')$, which means that in the coNP oracle one checks for all $q \in Q$ that $q(\hat{G}')$ does not contain $\tau_i(head(q_i))$. Again, this can either be done sequentially, for each

$q \in Q$ after the other, or in parallel by first guessing $q \in Q$ and then the corresponding homomorphism.

Towards the extension to UCQs$^\neg$, first note that there exist two possibilities why $Q_i(\hat{G}) \neq Q_i(\hat{G}')$: Either because of some $s \in Q_i(\hat{G})$ but $s \notin Q_i(\hat{G}')$, or because of some $s \in Q_i(\hat{G}')$ but $s \notin Q_i(\hat{G})$. This can be taken care of by guessing in step (3) which of the two cases applies, together with Q_i and q_i. Obviously, the verification of the guess in steps (4) and (5) needs to be adapted accordingly. Besides considering the additional guess on the kind of violation, step (5) must be further adapted to also treat UCQs$^\neg$ correctly. That is, for every guessed homomorphism τ from the body of some query q into some graph $\bar{G}$, instead of just checking whether $\tau(pos(q)) \subseteq \bar{G}$, it is also necessary to test that $\tau(neg(q)) \cap \bar{G} = \emptyset$. However, this additional check, just as the previous one, obviously fits into polynomial time. □

The corresponding hardness, presented in the next lemma, is based on the possibility to require homomorphisms of unrestricted size that give rise to solutions over the original graph to remain valid homomorphisms from the body of the query into the minimised graph. In the special case of the reduction presented, this will be used to enforce certain 3-colorings of a set of nodes to be preserved. Exploiting projection, this allows us to restrict the set of triples that may be removed from the graph. The basic idea of the reduction is the same as in the proof of Lemma 5.5. However, without constraints it is now no longer possible to enforce that allowed subsets of the RDF graph encode 3-colorings of some node set. Therefore the corresponding existential quantifier can no longer be expressed, which only allows for a reduction from Q-3COL$_{\forall,2}$.

Lemma 5.7. MINI-RDF$^{\subseteq,CQ}(G, \mathcal{R}, \mathcal{C}, \mathcal{Q})$, *when* $\mathcal{R}$ *is fixed,* $\mathcal{C} = \emptyset$, *and* $\mathcal{Q}$ *may contain arbitrary CQs is* Π_2^P*-hard, already if* $\mathcal{Q}$ *contains a single CQ only. The problem remains* Π_2^P*-hard, even if* $\mathcal{R} = \emptyset$ *and* G *is a fixed RDF graph.*

Proof. The proof is by reduction from Q-3COL$_{\forall,2}$. Hence let $\mathcal{G} = ((V, E), (V_1, V_2))$ be an arbitrary instance of Q-3COL$_{\forall,2}$ with $V = \{v_1, \dots, v_n\}$ and $|V_1| = p$. Define an instance $(G, \mathcal{R}, \mathcal{C}, \mathcal{Q})$ of MINI-RDF$^{\subseteq,CQ}$ with $\mathcal{R} = \mathcal{C} = \emptyset$ as follows. Let $G = G_{cols} \cup G_{e1} \cup G_{e2} \cup G_r$ be an RDF graph where

$$\begin{aligned}
G_{cols} &= \{0\ iscol\ 0\,.\,1\ iscol\ 1\,.\,2\ iscol\ 2\},\\
G_{e1} &= \{0\ e^*\ 0\,.\,1\ e^*\ 1\,.\,2\ e^*\ 2\,.\,0\ e^*\ 1\,.\\
&\qquad 0\ e^*\ 2\,.\,1\ e^*\ 2\,.\,1\ e^*\ 0\,.\,2\ e^*\ 0\,.\,2\ e^*\ 1\},\\
G_{e2} &= \{0\ e\ 2\,.\,0\ e\ 1\,.\,1\ e\ 2\,.\,1\ e\ 0\,.\,2\ e\ 0\,.\,2\ e\ 1\}, \text{ and}\\
G_r &= \{e\ s\ e\,.\,e^*\ s\ e^*\}.
\end{aligned}$$

Further, define $\mathcal{Q}$ containing a single query q as

$$\begin{aligned}
\mathcal{Q} = \{\ &\{C_i\ iscol\ C_i \mid v_i \in V\} \cup\\
&\{C_\alpha\ E\ C_\beta \mid (v_\alpha, v_\beta) \in E\} \cup\\
&\{X_1\ e\ X_2\,.\,X_1^*\ e^*\ X_2^*\} \cup\\
&\{E\ s\ E\,.\,e\ s\ e\}\\
&\qquad \rightarrow ans(X_1, X_2, X_1^*, X_2^*, C_{i_1}, \dots, C_{i_p})\}
\end{aligned}$$

where $C_1, \dots, C_n$ are new blank nodes for every $v_i \in V$, and $C_{i_1}, \dots, C_{i_p}$ are those $C_i \in \{C_1, \dots, C_n\}$ such that $v_i \in V_1$. This reduction is obviously feasible in LOGSPACE. To emphasize the main ideas of the reduction, it is convenient to consider the following two claims (the corresponding proofs are given in the appendix):

Claim 1: Consider G, and q as defined above together with the following sets: $\vec{X} = \{(x_1, x_2) \mid x_1, x_2 \in \{0, 1, 2\}, x_1 \neq x_2\}$, $\vec{X}^* = \{(x_1^*, x_2^*) \mid x_1^*, x_2^* \in \{0, 1, 2\}\}$, and $C = \{0, 1, 2\}^p$. Then $q(G) = \vec{X} \times \vec{X}^* \times C$.

Claim 2: Let $G' \subset G$ such that $q(G') = q(G)$. Then $G' = G \setminus \{e^*\ s\ e^*\}$.

Hence the intuition of the reduction is as follows: There exists a one-to-one correspondence between values on $C_1, \dots, C_n$ and colorings of V. If an answer can be retrieved by mappings E to e, then the values of $C_1, \dots, C_n$ encode a valid 3-coloring of (V, E). Hence if the triple $e^*\ s\ e^*$ is redundant, this means that for every possible coloring on V_1, there exists an extension to a valid 3-coloring of (V, E). In principle, the quantifier alternation is expressed exactly as in the proof of Lemma 5.5 except for the leading existential quantification, which cannot be expressed here. Hence, it only remains the possibility to say that *for all* mappings that return a required result tuple, *there exists* an extension to a homomorphism from the body of the query into the RDF graph. The remaining proof that the reduction indeed satisfies this intution can be found in the appendix. □

5.2.3. *Settings complete for* $\Delta_2^P[\log n]$

The main reason why (most of) the settings with head-b-bounded queries are computationally easier than the corresponding settings with arbitrary queries is that now the number of results of a query is polynomially bounded in the size of the active domain. In fact, note that for some query q, the set of possible results is $AD^{|head(q)|}$. This allows us, given some RDF graph G, a set of rules $\mathcal{R}$, and query q, to compute

$q(Cl_{\mathcal{R}}(G))$ in $\Delta_2^P[\log n]$. However, note that as long as $\mathcal{C} \neq \emptyset$, it is still necessary to check an exponential number of subgraphs $G' \subset G$. Hence computing for each of these subsets G' the result of $q(Cl_{\mathcal{R}}(G'))$ is not feasible in $\Delta_2^P[\log n]$. However, it turns out that for monotone queries $\mathcal{Q}$, it suffices to only compute $Cl_{\mathcal{R}}(G)$ explicitly, while this is not necessary for every subgraph $G' \subset G$. This is the main idea behind the proof of the following lemma. Since it only works for monotone queries, this is one of the results where there is no obvious way to extend the membership proof to UCQs$^{\neg}$. A complexity analysis of this setting for CQs$^{\neg}$ and UCQs$^{\neg}$ is left to future work.

Lemma 5.8. MINI-RDF$^{\subseteq,CQ}(G,\mathcal{R},\mathcal{C},\mathcal{Q})$, *for arbitrary $\mathcal{R}$, b-bounded $\mathcal{C}$, and where all CQs in $\mathcal{Q}$ have bounded head arity can be solved in* $\Delta_2^P[\log n]$.

Proof. The proof of the lemma is by showing that the following algorithm decides the problem in deterministic polynomial time with $O(\log n)$ calls to an NP-oracle:

```
1. For every $q \in \mathcal{Q}$, compute (and store) $q(\hat{G}) = q(Cl_{\mathcal{R}}(G))$
2. Test if there exists $G' \subset G$ that satisfies $\mathcal{C}$ and s.t. $q(Cl_{\mathcal{R}}(G')) = q(\hat{G})$.
```

Since the correctness of the algorithm follows immediately, it only remains to show that it indeed fits into the given time bound.

The *first step* can be computed as follows: For every $q \in Q$, test for every tuple $s \in AD^{|head(q)|}$ by a call to the NP-oracle if $s \in q(\hat{G})$. If this is the case, add s to $q(\hat{G})$. If s is in $q(\hat{G})$ can be decided in NP as follows:

```
1. Set $\hat{G}_0 = G$
2. For $j = 1, \dots, k = |AD|^3$:
   (a) Guess a rule $r\colon G_i \Rightarrow \{t_i\}$ from $\mathcal{R} \cup \{r_0\}$
   (b) Guess a mapping $\mu\colon G_i \to \hat{G}_{j-1}$
   (c) Check if $\mu$ is a homomorphism $G_i \to \hat{G}_{j-1}$
   (d) If it is, set $\hat{G}_j = \hat{G}_{j-1} \cup \{\mu(t_i)\}$, otherwise $\hat{G}_j = \hat{G}_{j-1}$
3. Guess an extension of the mapping defined as $\tau(head(q)) = s$ to $body(q) \to \hat{G}_k$
4. If $\tau(body(q)) \subseteq \hat{G}_k$, return "yes", otherwise return "no".
```

where r_0 mimics some "do nothing rule" $\emptyset \Rightarrow \emptyset$. The algorithm is obviously correct (just note that for some sequence of rule applications $\hat{G}_k = \hat{G}$ holds), and decides the problem in nondeterministic polynomial time.

Also the *second step* can be solved by a single call to an NP-oracle, namely by the following NP-algorithm:

```
1. Guess $G' \subset G$
2. Test if $G'$ satisfies $\mathcal{C}$, if not return "no".
3. Guess $\hat{G}' = Cl_{\mathcal{R}}(G')$
4. For every pair $(q, s)$ where $q \in \mathcal{Q}$ and $s \in q(\hat{G})$
   (a) For the homomorphism defined as $\tau(head(q)) = s$, guess an extension $\tau'\colon body(q) \to \hat{G}'$
   (b) Test if $\tau'(body(q)) \subseteq \hat{G}'$. If not, return "no"
5. Return "yes"
```

Step (3) summarises the steps that are laid out in more detail as step (2) in the previous algorithm. Again, the correctness of the algorithm is immediate.

Towards the $\Delta_2^P[\log n]$-membership, just note that since $\mathcal{Q}$ is assumed to be head-b-bounded, the size of $AD^{|head(q)|}$ is polynomial in the input. Since trivially $|\mathcal{Q}|$ is polynomially bounded in the input size, overall there is only a polynomial number of oracle calls of the first kind. Further, note that all these calls are non-adaptive. Hence the computation of $q(\hat{G})$ is in $\Delta_2^P[\log n]$, that is it can be done in deterministic polynomial time with $O(\log n)$ adaptive oracle calls. The second step consists of a single oracle call, which depends on the previous oracle calls. Now by the same arguments as in Lemma 4.3, this shows $\Delta_2^P[\log n]$-membership.

It remains to show that the result still holds for UCQs. This is achieved analogously to the extensions presented previously: Treating $Q \in \mathcal{Q}$ instead of CQs q is done by computing $Q(\hat{G})$ instead of $q(\hat{G})$ in the first step. This can be done by altering the first NP algorithm to guess $q \in Q$ together with τ. Also the NP algorithm for the second step can be easily altered by either guessing $q \in Q$ together with τ', or by iterating over all $q \in Q$. □

It is now easy to see why the algorithm presented above does not work for CQs$^{\neg}$ or UCQs$^{\neg}$. The "guess" of $Cl_{\mathcal{R}}(G')$ for $G' \subset G$ is not guaranteed to indeed return $Cl_{\mathcal{R}}(G')$, but might only find a subset of the closure. This is no problem for monotone queries, since every result retrieved over some subset of the closure will also be an answer over the closure. Obviously, this is not the case for nonmonotone queries. Finally, note that testing if such a "guess" indeed returns the closure would require a coNP-test.

Next, the two $\Delta_2^P[\log n]$-hardness results are presented. As already pointed out before, the reason for the need of the NP-oracle (instead of just a single NP algorithm), is that *all* answers to a query must be found, in order to be sure that the sets of answers indeed coincide. Now there are two reasons why this cannot be solved by a single NP algorithm. First, one must be sure that for some query q and RDF graph G, indeed all homomorphisms $\mathit{body}(q) \to G$ are found. Second, one must be sure that G is indeed the closure under some set of rules, that is that no more rule is applicable.

Hence the problem remains $\Delta_2^P[\log n]$-hard, as long as either deciding if some triple is a solution to a query or if some triple can be derived via some rule is NP-hard. This is shown next by reduction from ODD CLIQUE. The idea of both proofs is to define an instance of MINI-RDF$^{\subseteq,CQ}(G,\mathcal{R},\mathcal{C},\mathcal{Q})$ where the answers to some query $q \in \mathcal{Q}$ contain a special tuple if there exists a clique of even size i and one of size $i+1$. The test for the existence of such cliques is encoded either in the bodies of the queries or rules, depending on which of those are allowed to be unbounded. In the first case, the answer tuple is created directly while in the second case a rule application creates an intermediate triple which is then output by a simple query. Further, G is defined in such a way that it contains a certain triple that allows the corresponding rules or queries to fire whenever there exists a clique of even size i, independent of the existence of a clique of size $i+1$. Finally, it is made sure that if any other triple except for this special triple is removed from G, then $\mathcal{Q}$ cannot return the same answers over the corresponding subgraph as over G. Hence there exists a subgraph of G that allows one to derive the same answers to $\mathcal{Q}$ as over G iff this tuple is redundant which in turn is the case iff the size of the biggest clique is odd.

The first hardness proof exploits the case that the NP-hard test for the existence of a clique of a certain size can be encoded in the bodies of the queries.

Lemma 5.9. MINI-RDF$^{\subseteq,CQ}(G,\mathcal{R},\mathcal{C},\mathcal{Q})$, *when* $\mathcal{R}$ *is fixed,* $\mathcal{C} = \emptyset$, *and the CQs in* $\mathcal{Q}$ *may have arbitrary BGPs in their bodies but have bounded head arity, is* $\Delta_2^P[\log n]$*-hard. The problem remains* $\Delta_2^P[\log n]$*-hard even if* $\mathcal{R} = \emptyset$.

Proof. The proof is by reduction from ODD CLIQUE. Let (V,E) be an arbitrary instance of ODD CLIQUE with $V = \{v_1,\dots,v_n\}$. Define an instance $(G,\mathcal{R},\mathcal{C},\mathcal{Q})$ of MINI-RDF$^{\subseteq,CQ}(G,\mathcal{R},\mathcal{C},\mathcal{Q})$ with $\mathcal{C} = \mathcal{R} = \emptyset$ as follows. Let $G = G_{e1} \cup G_{e2}$ where

$$\begin{aligned}
G_{e1} = \{v_\alpha\ e\ v_\beta \mid {}& (v_i,v_j) \in E,\\
& \alpha = \min(i,j), \beta = \max(i,j)\},\\
G_{e2} = \{v_\alpha\ e^*\ v_\beta \mid {}& (v_i,v_j) \in E,\\
& \alpha = \min(i,j), \beta = \max(i,j)\} \cup\\
& \{0\ e^*\ 0\},
\end{aligned}$$

and a new URI v_i is introduced for every node $v_i \in V$ (by slight abuse of notation, v_i is used to denote both, URIs in G and nodes in V). Further, $\mathcal{Q}$ contains the following queries: For every $i \in \{2,\dots,n\}$, if $i \bmod 2 = 0$ then
$q_i = \{X_r\ e\ X_s \mid 1 \le r < s \le i\} \cup$
$\{X'_r\ e^*\ X'_s \mid 1 \le r < s \le i+1\} \to \mathit{ans}(o_{i+1})$.
Then $\mathcal{Q}$ is defined as

$$\mathcal{Q} = \{q_i \mid 1 \le i \le n\} \cup \{\{Y_1\ e\ Y_2\,.\,Y_1\ e^*\ Y_2\} \to \mathit{ans}(Y_1,Y_2)\}$$

The reduction is obviously feasible in LOGSPACE. It is convenient to summarise the main properties of the instance of MINI-RDF$^{\subseteq,CQ}$ defined above in the following claims. Their correctness is shown in the appendix:

Claim 1: Let $G' \subset G$ such that $q(G) = q(G')$ for every $q \in \mathcal{Q}$. Then $G' = G \setminus \{0\ e^*\ 0\}$.

Claim 2: Let m be the size of the biggest clique in (V,E). If m is odd, then $q_i(G) = \{(o_{i+1})\}$ for $i \in \{2,4,\dots,m-1\}$ and $q_i(G) = \emptyset$ for $i \in \{m+1,m+3,\dots,n\}$. If m is even, then $q_i(G) = \{(o_{i+1})\}$ for $i \in \{2,4,\dots,m\}$ and $q_i(G) = \emptyset$ for $i \in \{m+2,m+4,\dots,n\}$.

From these claims, it is easy to see that the intuition of the reduction is as described above: If the size of the biggest clique in (V,E) is odd, then there still exists a homomorphism from the body of the query q_{m-1} into $G' = G \setminus \{0\ e^*\ 0\}$, hence $q_{m-1}(G') = \{o_m\}$. On the other hand, if m, the size of the biggest clique, is even, then there does no longer exist a homomorphism from the body of q_m into G', and therefore $q_m(G') = \emptyset$, while $q_m(G) = \{o_{m+1}\}$. A formal proof of this property is given in the appendix. □

The next hardness result exploits the second reason for $\Delta_2^P[\log n]$-hardness mentioned above. In the setting considered in the next lemma, the set of answers can be computed efficiently, but testing if some rule is applicable is hard. Hence in the proof of the lemma, the test for the existence of a clique of a certain size is encoded in the bodies of the rules, instead of the bodies of the queries.

Lemma 5.10. MINI-RDF$^{\subseteq,CQ}(G,\mathcal{R},\mathcal{C},\mathcal{Q})$, *when* $\mathcal{C}=\emptyset$, $\mathcal{Q}$ *contains only body-b-bounded CQs but* $\mathcal{R}$ *may contain arbitrary rules is* $\Delta_2^P[\log n]$*-hard. It remains* $\Delta_2^P[\log n]$*-hard even if* $\mathcal{Q}$ *contains a single CQ only.*

Proof. The proof is by reduction from ODD CLIQUE, hence let (V,E) be an arbitrary instance of ODD CLIQUE with $V=\{v_1,\dots,v_n\}$ and assume without loss of generality that $E\neq\emptyset$. Define an instance $(G,\mathcal{R},\mathcal{C},\mathcal{Q})$ of MINI-RDF$^{\subseteq,CQ}(G,\mathcal{R},\mathcal{C},\mathcal{Q})$ as follows. Let $G=G_{ord}\cup G_e\cup G_c$ where

$$\begin{aligned}G_{ord}&=\{i\ succ\ i+1\mid 1\le i\le n, i \mod 2=0\},\\ G_e&=\{v_\alpha\ e\ v_\beta\mid (v_i,v_j)\in E,\\ &\qquad\qquad \alpha=\min(i,j),\beta=\max(i,j)\},\\ G_c&=\{c\ c\ c\},\end{aligned}$$

and a new URI v_i is introduced for every node $v_i\in V$ (by slight abuse of notation, v_i is used to denote both, nodes in V and URIs in G). Next, for $i\in\{2,\dots,n\}$ consider the following rules r_i. If $i \mod 2=1$, then
$r_i=\{X_r\ e\ X_s\mid 1\le r<s\le i\}\Rightarrow\{i\ clique\ i\}$
and if $i \mod 2=0$ then
$r_i=\{X_r\ e\ X_s\mid 1\le r<s\le i\}\cup\{c\ c\ c\}$
$\Rightarrow\{i\ clique\ i\}$
Using these rules, define $\mathcal{R}$ as

$$\begin{aligned}\mathcal{R}=\{&r_i\mid 1\le i\le n\}\cup\\ &\{\{X\ succ\ Y\,.\,Y\ clique\ Y\}\Rightarrow\{X\ clique\ X\}\}\end{aligned}$$

and finally, containing a single query q let $\mathcal{Q}$ be defined as

$$\begin{aligned}\mathcal{Q}=\{\{&X\ succ\ Y\,.\,V_1\ e\ V_2\,.\,C\ clique\ C\}\\ &\to ans(X,Y,V_1,V_2,C)\}\end{aligned}$$

This reduction is obviously feasible in LOGSPACE. It is again convenient to explicitly state two major properties of this reduction in the following claims, which are proven in the appendix.

Claim 1: Let m be the size of the biggest clique in (V,E). Then $Cl_{\mathcal{R}}(G)=G\cup\{i\ clique\ i\mid 2\le i\le m\}$.

Claim 2: Assume $G'\subset G$ such that $q(Cl_{\mathcal{R}}(G'))=q(Cl_{\mathcal{R}}(G))$. Then $G\setminus G'=Cl_{\mathcal{R}}(G)\setminus Cl_{\mathcal{R}}(G')=\{c\ c\ c\}$.

Concerning the intuition of the reduction, it follows from Claim 1 that over G for every clique in (V,E) of size i a triple $i\ clique\ i$ is created by rule r_i. However, over $G'=G\setminus\{c\ c\ c\}$ (by Claim 2 the subgraph of G of interest), the triple $i\ clique\ i$ can only be derived from rule r_i for odd values i. Now the additional rule in $\mathcal{R}$ creates a triple $j\ clique\ j$ only if there exists a triple $j+1\ clique\ j+1$. Hence if the size of the biggest clique in (V,E) is odd, then still all triples $i\ clique\ i$ can be derived. However, if the size m of the biggest clique is even, then the triple $m\ clique\ m$ cannot be derived. A formal proof of this property (and hence of the correctness of the reduction) is given in the appendix. □

Note that in the above case, the hardness is due to the fact that the closure of the given graph must be computed. To see that in the last setting, this was indeed the source for the $\Delta_2^P[\log n]$-hardness, consider a slight variation of the MINI-RDF$^{\subseteq,CQ}(G,\mathcal{R},\mathcal{C},\mathcal{Q})$ problem: Assume that it is guaranteed that $G=Cl_{\mathcal{R}}(G)$. Then it is easy to see that the problem remains in NP, even if $\mathcal{R}$ is allowed to contain arbitrary rules, $\mathcal{C}$ is restricted to b-bounded constraints and $\mathcal{Q}$ to body-b-bounded UCQs.

5.2.4. *Settings complete for* NP *and tractable cases*

It was already discussed above that only in settings where both problems, computing the closure and computing the set of answers to a query over some RDF graph are tractable, MINI-RDF$^{\subseteq,CQ}$ is no longer $\Delta_2^P[\log n]$-hard. In fact, it can be quite easily seen that in this case, the problem is in NP.

Lemma 5.11. MINI-RDF$^{\subseteq,CQ}(G,\mathcal{R},\mathcal{C},\mathcal{Q})$, *where* $\mathcal{R}$ *and* $\mathcal{C}$ *are b-bounded and* $\mathcal{Q}$ *is body-b-bounded can be solved in* NP. *The problem remains in* NP *even if* $\mathcal{Q}$ *is a set of body-b-bounded UCQs*$^\neg$.

Proof. The problem can be decided by the following nondeterministic algorithm:

1. *Compute* $\hat{G}=Cl_{\mathcal{R}}(G)$
2. *Guess a subset* $G'\subset G$
3. *Check if* G' *satisfies* $\mathcal{C}$ *(if not, return "no")*
4. *Compute* $\hat{G'}=Cl_{\mathcal{R}}(G')$
5. *For every* $q\in\mathcal{Q}$*, test if* $q(\hat{G})=q(\hat{G'})$

The correctness of the algorithm follows immediately. Its polynomial runtime stems from the following facts: By [19, Proposition 9], $Cl_{\mathcal{R}}(G)$ (and therefore also $Cl_{\mathcal{R}}(G')$) can be computed in polynomial time, and step (3) according to [19, Proposition 3]. Step (2) introduces the nondeterminism, and step (5) can be solved even for UCQs$^\neg$ by performing the following steps for each $Q\in\mathcal{Q}$:

1. *Set* $A=A'=\emptyset$
2. *For every* $q\in Q$ *and every homomorphism* $\tau\colon pos(q)\to\hat{G}$

```
        (a) If for every $t \in \tau(neg(q))$: $t \notin \hat{G}$: then set
            $A = A \cup \{\tau(head(q))\}$
    3. For every $q \in Q$ and every homomorphism
       $\tau\colon pos(q) \to \hat{G}'$
        (a) If for every $t \in \tau(neg(q))$: $t \notin \hat{G}'$: then set
            $A' = A' \cup \{\tau(head(q))\}$
    4. If $A \neq A'$ return "no"
    5. Test next $Q$. If none is left, return "yes"
```

The correctness follows immediately from the semantics of UCQs$^\neg$. To see that this is indeed feasible in polynomial time, note that the maximal number of homomorphisms to be checked in steps (2) and (3) for every $q \in Q$ is $|AD|^{3*b}$, since the number of triples in $body(q)$ is bounded by some constant b. □

Note that since in the above setting both, the closure under $\mathcal{R}$ and the answers to $\mathcal{Q}$ can be computed efficiently, membership even holds for UCQs$^\neg$. That is, monotone queries are no longer necessary.

Since for these settings the complexity is not higher than for MINI-RDF$^\subseteq$, the hardness follows immediately from Proposition 5.2.

Corollary 5.12. MINI-RDF$^{\subseteq,CQ}(G,\mathcal{R},\mathcal{C},\mathcal{Q})$*, when both, $\mathcal{R}$ and $\mathcal{C}$ are fixed and $\mathcal{Q}$ contains only body-b-bounded CQs is* NP*-hard, already if $\mathcal{Q}$ contains a single CQ only.*

The treatment of MINI-RDF$^\subseteq(G,\mathcal{R},\mathcal{C})$ is concluded with a tractable case. Just as in Section 3, if $\mathcal{C} = \emptyset$ it suffices to check only a polynomial number of subgraphs $G' \subset G$, namely all those graphs missing exactly one triple from G.

Lemma 5.13. MINI-RDF$^{\subseteq,CQ}(G,\mathcal{R},\mathcal{C},\mathcal{Q})$*, for b-bounded $\mathcal{R}$, $\mathcal{C} = \emptyset$, and body-b-bounded $\mathcal{Q}$ can be solved in* PTIME*. The problem remains in* PTIME *even if $\mathcal{Q}$ is a set of body-b-bounded UCQs$^\neg$.*

Proof. The lemma follows immediately from the proof of Lemma 5.11. The problem can be solved by almost the same algorithm that was presented there, only that it is not necessary in step (2) to *"guess a subset"*, but instead it suffices to check all subsets of G missing exactly one triple from G. □

5.3. Rule minimisation

Like for the problem of graph minimisation, also for the problem RDF-RULEMIN$^{\subseteq,CQ}$ the hardness results from Section 4 carry over. The reason is again that RDF-RULEMIN$^\subseteq(G,\mathcal{R})$ is a special case of RDF-RULEMIN$^{\subseteq,CQ}(G,\mathcal{R},\mathcal{Q})$. However, this can be used in a single proof in this section only. All other results have to be shown explicitly.

Lemma 5.14. RDF-RULEMIN$^{\subseteq,CQ}(G,\mathcal{R},\mathcal{Q})$*, for arbitrary $\mathcal{R}$ and where $\mathcal{Q}$ may contain arbitrary CQs can be solved in Π_2^P. The problem remains in Π_2^P even if $\mathcal{Q}$ is a set of arbitrary UCQs$^\neg$.*

Proof. This result is proven by devising a nondeterministic algorithm that, using a coNP-oracle, decides the co-problem in polynomial time. That is, it shows that the co-problem, hence deciding if $\mathcal{R}$ is already minimal, is in Σ_2^P.

```
    1. Compute $\hat{G} = Cl_{\mathcal{R}}(G)$
    2. For every $r \in \mathcal{R}$:
        (a) Let $\mathcal{R}' = \mathcal{R} \setminus \{r\}$
        (b) Compute $\hat{G}' = Cl_{\mathcal{R}'}(G)$
        (c) Guess $q \in \mathcal{Q}$ and homomorphism
            $\tau\colon body(q) \to \hat{G}$
        (d) Test if $\tau(head(q)) \notin q(\hat{G}')$
        (e) If $\tau(head(q)) \in q(\hat{G}')$, return "no"
    3. Return "yes"
```

Hence the algorithm proceeds as follows: First it computes the closure of G under $\mathcal{R}$. Then it verifies that whatever rule $r \in \mathcal{R}$ is dropped, there exists some query $q \in Q$ such that at least one triple from $q(Cl_{\mathcal{R}}(G))$ is not in the answer of q over the closure of G under $\mathcal{R} \setminus \{r\}$. It follows immediately that the algorithm is indeed correct. Further, it is easy to see that it runs in polynomial time on a nondeterministic Turing machine with access to a coNP oracle. The only step for which this is not trivial is step (2d). However, this can be solved by a coNP oracle as follows:

```
    1. Test for every mapping $\tau'\colon body(q) \to \hat{G}'$ with
       $\tau'(body(q)) \subseteq \hat{G}'$, if $\tau'(head(q)) \neq \tau(head(q))$
```

For how to compute the closure in Δ_2^P (even $\Delta_2^P[\log n]$), see for example the proof of Lemma 4.3.

Finally, the extension to UCQs$^\neg$ is analogous to the extension presented in the proof of Lemma 5.6. □

The corresponding hardness proof, given next, is very similar to the proof of Lemma 5.7. Recall that the proof of Lemma 5.7 was by reduction from Q-3COL$_{\forall,2}$. Given an instance $((V,E),(V_1,V_2))$, the basic idea of the reduction was to define an RDF graph G and a query q, such that $q(G)$ contained an encoding for every possible coloring of V_1. The remaining instance was defined in such a way that $q(G')$ (for "valid" $G' \subset G$) was able to return the same answer only if each such coloring can be extended to a valid 3-coloring of (V, E). The idea of the following proof is the same. The only difference is that now $q(G)$ returns only encodings of those colorings of V_1 that can be extended to valid 3-colorings of (V, E). In addition, $\mathcal{R}$ contains a single rule such that $q(Cl_{\mathcal{R}}(G))$ contains encodings of all possible colorings of V_1. Hence this rule is only redundant iff all possible colorings V_1 can be extended to valid 3-colorings of (V, E). The formal proof is given in the appendix.

Lemma 5.15. RDF-RULEMIN$^{\subseteq,CQ}(G, \mathcal{R}, \mathcal{Q})$*, where $\mathcal{R}$ may contain arbitrary rules and $\mathcal{Q}$ may contain arbitrary CQs, is Π_2^P-hard. The problem remains Π_2^P-hard, even if $\mathcal{R}$ and $\mathcal{Q}$ each contain a single element only.*

Note that the following membership result even holds in the presence of nonmonotone queries. The reason for this is that to solve the rule minimisation problem, it suffices to check only a polynomial number of subsets. Hence it is feasible to first compute the closures under all these sets of rules, and then to explicitly compute the set of answers over these closures (since the queries are head-b-bounded, there are at most polynomially many answers to each query).

Lemma 5.16. RDF-RULEMIN$^{\subseteq,CQ}(G, \mathcal{R}, \mathcal{Q})$*, for arbitrary rules $\mathcal{R}$ and head-b-bounded CQs $\mathcal{Q}$, can be solved in $\Delta_2^P[\log n]$. The problem remains in $\Delta_2^P[\log n]$ even if $\mathcal{Q}$ contains UCQs$^\neg$.*

Proof. The proof is by devising a deterministic algorithm that solves the problem in polynomial time with at most $O(\log n)$ calls to an NP oracle. In the following, let $\mathcal{R} = \{r_1, \ldots, r_n\}$, and for $i \in [n]$ let $\mathcal{R}_i = \mathcal{R} \setminus \{r_i\}$.

1. *For every pair $(q, \mathcal{R}')$ of $q \in Q$ and $\mathcal{R}' \in \{\mathcal{R}, \mathcal{R}_1, \ldots, \mathcal{R}_n\}$, compute (and store) $q(Cl_{\mathcal{R}'}(G))$*
2. *Check if there exists $\mathcal{R}_i$ for $i \in [n]$ s.t. for all $q \in Q$: $q(Cl_{R_i}(G)) = q(Cl_{\mathcal{R}}(G))$*

The correctness of the algorithm follows immediately. Further, given the result of step (1), step (2) can be decided in deterministic polynomial time. Hence it remains to show that step (1) can be solved in polynomial time with at most $O(\log n)$ calls to an NP oracle. To see that this is indeed the case, divide the step into two sub-steps: First, compute $Cl_{\mathcal{R}'}(G)$ for every $\mathcal{R}' \in \{\mathcal{R}, \mathcal{R}_1, \ldots, \mathcal{R}_n\}$. This is feasible in $\mathrm{P}_{\|}^{\mathrm{NP}}$(cf. proof of Lemma 4.3). Next, given all these closures, $q(Cl_{\mathcal{R}'}(G))$ can be computed by deciding for each $s \in AD^{|head(q)|}$ by a call to an NP oracle if $s \in q(Cl_{\mathcal{R}'}(G))$ (this can be decided by a simple "guess and check" algorithm). This again requires only a polynomial number of non-adaptive oracle calls (although they depend on the oracle calls from the first sub-step). Hence, both sub-steps of step (1) can be solved in deterministic polynomial time with at most $O(\log n)$ calls to an NP oracle. Therefore also the complete step (1) can be solved within this time bound. □

Note that the same arguments used in the above proof also allow one to show that MINI-RDF$^{\subseteq,CQ}$ can be solved in $\Delta_2^P[\log n]$ if $\mathcal{R}$ contains arbitrary rules, $\mathcal{C} = \emptyset$ and $\mathcal{Q}$ is a set of body-b-bounded UCQs$^\neg$.

Similarly to the graph minimisation problem, there are again two sources for the $\Delta_2^P[\log n]$-hardness: On the one hand it must be made sure that given some RDF graph and a query, indeed *all* answers to a query over this graph are computed. On the other hand, given some RDF graph and a set of rules, it must be assured that all rules were applied exhaustively before the query answers are computed. The next hardness result considers the case where computing the answers to the queries is hard.

Lemma 5.17. RDF-RULEMIN$^{\subseteq,CQ}(G, \mathcal{R}, \mathcal{Q})$ *is $\Delta_2^P[\log n]$-hard if all rules in $\mathcal{R}$ are b-bounded and $\mathcal{Q}$ contains head-b-bounded CQs. The problem remains $\Delta_2^P[\log n]$-hard, even if $\mathcal{R}$ contains a single, fixed rule.*

Proof. The proof, done by reduction from the problem ODD CLIQUE, is a slight variation of the proof of Lemma 5.9. Given an arbitrary instance of ODD CLIQUE, denote with $(G', \emptyset, \emptyset, \mathcal{Q}')$ the instance of MINI-RDF$^{\subseteq,CQ}$ defined by the reduction presented there. Recall the basic idea of the reduction, which was that the only triple $t \in G'$ that might be redundant was $t = 0\ e^*\ 0$. Hence define an instance $(G, \mathcal{R}, \mathcal{Q})$ of RDF-RULEMIN$^{\subseteq,CQ}(G, \mathcal{R}, \mathcal{Q})$ as $G = G' \setminus \{t\}$, $\mathcal{R} = \{\{S\ P\ O\} \Rightarrow \{0\ e^*\ 0\}\}$, and $\mathcal{Q} = \mathcal{Q}'$. I.e. instead of the triple t, now the only existing rule is redundant iff the size of the biggest clique in the instance of ODD CLIQUE is odd. The correctness follows immediately from the correctness of the reduction in the proof of Lemma 5.9. □

The following hardness result, which follows immediately from Lemma 4.3, considers the case where the set of answers over an RDF graph can be computed efficiently, but it is intractable to compute the closure of a graph with respect to a set of rules.

Lemma 5.18. RDF-RULEMIN$^{\subseteq,CQ}(G,\mathcal{R},\mathcal{Q})$ *is* $\Delta_2^P[\log n]$*-hard if* $\mathcal{R}$ *contains arbitrary rules and* $\mathcal{Q}$ *is restricted to body-b-bounded CQs. The problem remains* $\Delta_2^P[\log n]$*-hard, even if* $\mathcal{Q}$ *contains a single CQ only.*

Proof. To prove this result, note that $(G,\mathcal{R},\mathcal{C})$ is a positive instance of RDF-RULEMIN$^{\subseteq}(G,\mathcal{R})$ if and only if $(G,\mathcal{R},\mathcal{C},\mathcal{Q})$ is a positive instance of RDF-RULEMIN$^{\subseteq,CQ}(G,\mathcal{R},\mathcal{Q})$, where $\mathcal{Q}=\{q\}$ with $q=\{S\ P\ O\}\rightarrow ans(S,P,O)$. To see that this is indeed the case, just note that $q(Cl_{\mathcal{R}}(G))=q(Cl_{\mathcal{R}'}(G))$ iff $Cl_{\mathcal{R}}(G)=Cl_{\mathcal{R}'}(G)$, for every $\mathcal{R}'\subset\mathcal{R}$.

The lemma thus follows from Lemma 4.3. □

The two results above show that for the problem RDF-RULEMIN$^{\subseteq,CQ}$, the situation is very similar to MINI-RDF$^{\subseteq,CQ}$: As long as either computing the closure or the set of query answers is hard, the problem is $\Delta_2^P[\log n]$-hard. The next result shows yet another similarity between these two problems: If both, the closure and the query answers can be computed efficiently, and if it suffices to check a polynomial number of subsets (which is always the case for RDF-RULEMIN$^{\subseteq,CQ}$), then the problem becomes tractable.

Lemma 5.19. RDF-RULEMIN$^{\subseteq,CQ}(G,\mathcal{R},\mathcal{Q})$ *where* $\mathcal{R}$ *is a set of b-bounded rules and* $\mathcal{Q}$ *is a set of body-b-bounded CQs can be decided in* PTIME. *The problem even remains in* PTIME *if* $\mathcal{Q}$ *is a set of body-b-bounded UCQs*$^{\neg}$.

Proof. The problem can be decided by the following deterministic adaption of the nondeterministic algorithm presented in the proof of Lemma 5.14.

```
1. Compute $\hat{G} = Cl_{\mathcal{R}}(G)$
2. For every $r \in \mathcal{R}$:
   (a) Let $\mathcal{R}' = \mathcal{R} \setminus \{r\}$
   (b) Compute $\hat{G}' = Cl_{\mathcal{R}'}(G)$
   (c) For every $q \in \mathcal{Q}$, compute $q(\hat{G})$ and $q(\hat{G}')$,
       and check if they are the same
   (d) If this holds for all $q \in \mathcal{Q}$, return "yes", otherwise try next $r \in \mathcal{R}$
3. Return "no"
```

The correctness of the algorithm follows immediately. Just note that unlike the proof of Lemma 5.14, the above algorithm solves RDF-RULEMIN$^{\subseteq,CQ}$ directly, and not the co-problem. Its polynomial runtime, and also the extension to UCQs$^{\neg}$ follows for the same reason as in Lemma 5.13 and Lemma 5.11, respectively. □

5.4. Beyond conjunctive queries – SPARQL

RDF minimization w.r.t. (unions of) conjunctive queries can be extended to more expressive query languages. For example, some of the results above were shown to hold even if the (U)CQs are allowed to contain negation in the body. However, this was not the case in all of the settings considered. Although a deeper study of settings with UCQs$^{\neg}$ is left to future work, it is to be expected that allowing for negation increases the complexity of the problem (but not exceeding Σ_3^P).

In general, it is to be expected that more expressive query languages also increase the computational complexity of the problems under consideration. For example, when allowing non-recursive datalog queries with negation (a query language which covers all of SPARQL [2]), then it can be easily seen that the complexity of the problems considered here will be dominated by the complexity of query evaluation, which is PSPACE-complete in this case.

The goal of this section is to initiate the study of RDF minimisation w.r.t. *SPARQL*. Note das UCQs as studied so far, but without projection (i.e. if adding the requirement that all variables in the body of a CQ also occur in its head) correspond to SPARQL graph patterns consisting only of SPARQL triple patterns, AND, and UNION. Further, CQs (again without projection) correspond to the fragment of SPARQL graph patterns built from SPARQL triple patterns using AND only. Note however that projection was an important tool in most of the hardness proofs that made use of the set of queries. Hence, the results presented in this paper so far carry over to SPARQL queries built from those kind of SPARQL graph patterns and projection. In addition, in Section 2 the body of CQs was defined to be a FBGP. Hence those types of CQs also cover certain SPARQL filter expressions, namely those testing if a variable is bound to an uri, a literal or a blank node, respectively. Hence the results immediately extend to settings allowing for SPARQL queries built from those SPARQL graph patterns. Further, the results can be easily shown to hold also when allowing for equality as a filter expression.

However, when allowing in addition for the OPT-operator, recall that given a SPARQL graph pattern P and a mapping μ, deciding if $\mu\in[\![P]\!]_G$ holds is PSPACE-complete [24]. The previous results therefore do not extend to arbitrary SPARQL graph patterns. However, the OPT-operator implements a main feature of SPARQL, namely to derive useful results even in the case that not all parts of the query can be answered. This is of great importance when querying the web where incomplete data sources are common. Hence, arbitrary SPARQL graph patterns are considered next. This gives the following result.

Theorem 5.20. *Let G be an RDF graph, $\mathcal{C}$ a set of tgds, and $\mathcal{P}$ a set of SPARQL graph patterns. Further, assume that G satisfies $\mathcal{C}$. Then deciding if there exists $G' \subset G$ such that G' satisfies $\mathcal{C}$ and $[\![P]\!]_G = [\![P]\!]_{G'}$ holds for all $P \in \mathcal{P}$ is* PSPACE*-complete.*

The problem remains PSPACE*-hard, even if all patterns in $\mathcal{P}$ are constructed using* AND, UNION, *and* OPT *only.*

Proof. Membership is shown by the following algorithm:

1. *Guess a subgraph $G' \subset G$*
2. *Check if G' satisfies $\mathcal{C}$*
3. *For every $P \in \mathcal{P}$:*
   (a) For every $\mu \in [\![P]\!]_G$, test if $\mu \in [\![P]\!]_{G'}$
   (b) For every $\mu \in [\![P]\!]_{G'}$, test if $\mu \in [\![P]\!]_G$

The correctness of this algorithm follows immediately. To see that it indeed solves the problem in PSPACE, just note that both, G' and each μ is of polynomial size.

For the *hardness*, consider the proof of [24, Theorem 3.4]. This proof is by reduction from QSAT, that is the problem of deciding if an arbitrary quantified Boolean formula is valid, to the following problem:

– Given an RDF graph G, a graph pattern P and a mapping μ, decide if $\mu \in [\![P]\!]_G$.

Given an arbitrary quantified Boolean formula ϕ, the reduction in [24] shows how to define a graph pattern P_ϕ such that $\mu \in [\![P_\phi]\!]_G$ iff ϕ is valid. Thereby $G = \{a\ tv\ 0 \,.\, a\ tv\ 1 \,.\, a\ \mathit{false}\ 0 \,.\, a\ \mathit{true}\ 1\}$, and μ is defined on a variable B_0 only, with $\mu(B_0) = 1$. Further, P_ϕ consists of AND, UNION, and OPT only.

Now define a reduction from this problem to the problem mentioned in the theorem as follows: Define an RDF graph $\hat{G}$ as

$$\hat{G} = G \cup \{1\ x\ 1 \,.\, c\ c\ c\}.$$

Further, define the set $\mathcal{C}$ of constraints as

$$\begin{aligned}\mathcal{C} = \{&\{c\ c\ c\} \Rightarrow \hat{G};\\ &\{1\ x\ 1\} \Rightarrow \{c\ c\ c\};\\ &\{S\ P\ O\} \Rightarrow G\}\end{aligned}$$

Finally, define a SPARQL graph pattern Q as

$$Q = (B_0\ x\ B_0\ \text{ AND }\ c\ c\ c)\ \text{ UNION }\ P_\phi.$$

It can now be easily checked that $\mu \in [\![Q]\!]_{\hat{G}}$. Also, an inspection of the reduction in [24] shows that the additional two triples do not have any effect on the answers of P_ϕ. Further, the only subgraph $G' \subset \hat{G}$ that satisfies $\mathcal{C}$ is G. Since $[\![(B_0\ x\ B_0\ \text{AND}\ c\ c\ c)]\!]_G = \emptyset$ and $[\![P_\phi]\!]_G = [\![P_\phi]\!]_{G'}$, it follows that $[\![Q]\!]_{G'=G} = [\![Q]\!]_{\hat{G}}$ iff $\mu \in [\![Q]\!]_G$, which concludes the proof. □

Note that the previous result does not mention any rules, in order to keep the discussion short. However, it is easy to see that the algorithm can be adapted to additionally compute the closures of G and G' under a given set of rules without changing its complexity.

In [24], it was shown that the high complexity of evaluating SPARQL graph patterns arises from an unrestricted use of the OPT-operator. In addition, the same reason may lead to unintuitive results. As a solution, the fragment of *well-designed* SPARQL patterns was defiend in [24], that are defined as follows. First of all, note that every SPARQL graph pattern P is equivalent to some pattern of the form $(P_1\ \text{UNION}\ P_2\ \text{UNION}\ \dots\ \text{UNION}\ P_n)$ where each P_i ($i \in [n]$) is UNION-free [24, Proposition 3.8]. Patterns of this form are said to be in UNION-normal form. Further, a UNION-free pattern P is well-designed if on the one hand for every subpattern $(Q\ \text{FILTER}\ R)$ of P it holds that all variables occuring in R occur also in Q, and on the other hand for every subpattern $P' = (P_1\ \text{OPT}\ P_2)$ of P it holds that every variable X that occurs both inside P_2 and outside P' also occurs in P_1. Finally a pattern in UNION-normal form is well-designed if each of the UNION-free patterns $P_1, \dots, P_n$ is well-designed. It was shown that for a well-designed SPARQL graph pattern P and a mapping μ, testing if $\mu \in [\![P]\!]_G$ is coNP-complete [24], hence significantly lower than for arbitrary SPARQL graph patterns. The following result shows that also the complexity of RDF minimisation decreases in the presence of well-designed SPARQL graph patterns.

Theorem 5.21. *Let G be an RDF graph, $\mathcal{C}$ a set of b-bounded tgds, and $\mathcal{P}$ a set of well-designed SPARQL graph patterns in* UNION *normal form. Further, assume that G satisfies $\mathcal{C}$. Then deciding if there exists $G' \subset G$ such that G' satisfies $\mathcal{C}$ and $[\![P]\!]_G = [\![P]\!]_{G'}$ holds for all $P \in \mathcal{P}$ is in Σ_3^P and Σ_2^P-hard.*

Proof. The Σ_3^P-membership is due to the following algorithm:

1. *Guess a subgraph $G' \subset G$*
2. *Test if G' satisfies $\mathcal{C}$*
3. *Test if $[\![P]\!]_G = [\![P]\!]_{G'}$*

The correctness of the algorithm is immediate. To see that it indeed decides the problem in Σ_3^P, note that step (3) can be decided by a Π_2^P-oracle, i.e., testing if $[\![P]\!]_G \neq [\![P]\!]_{G'}$ is in Σ_2^P: by [24, Theorem 4.9], for this class of SPARQL graph patterns, deciding if some mapping $\mu \in [\![P]\!]_G$ is coNP-complete. Hence to test $[\![P]\!]_G \neq [\![P]\!]_{G'}$, we first guess μ, and then decide by two calls to an NP oracle if μ is contained in $[\![P]\!]_G$ but not in $[\![P]\!]_{G'}$.

For the *hardness*, consider the following reduction from Q-3COL$_{\exists,2}$. Let $\mathcal{G} = ((V,E),(V_1,V_2))$ be an arbitrary instance of Q-3COL$_{\exists,2}$ with $V = \{v_1,\dots,v_n\}$. Then define an RDF graph G as $G = G_{cols} \cup G_e \cup G_{v1} \cup G_{col1} \cup G_b$ where

$$\begin{aligned}
G_{cols} &= \{0\ iscol\ 0\,.\,1\ iscol\ 1\,.\,1\ iscol\ 1\},\\
G_e &= \{0\ e\ 1\,.\,0\ e\ 2\,.\,1\ e\ 0\,.\,1\ e\ 2\,.\,2\ e\ 0\,.\,2\ e\ 1\},\\
G_{v1} &= \{v_i\ v\ v_i \mid v_i \in V_1\},\\
G_{col1} &= \{v_i\ a\ 0\,.\,v_i\ a\ 1\,.\,v_i\ a\ 2\,.\ \mid v_i \in V_1\},\\
G_b &= \{b\ b\ b\},
\end{aligned}$$

and each v_i is a new URI for each $v_i \in V_1$ (where by slight abuse of notation, v_i is used to denote both, URIs in G and nodes in V). Further, define $\mathcal{C}$ as $\mathcal{C} = \mathcal{C}_{basic} \cup \mathcal{C}_{col1} \cup \mathcal{C}_0 \cup \mathcal{C}_1$ where

$$\begin{aligned}
\mathcal{C}_{basic} &= \{\{S\ P\ O\} \Rightarrow G_{cols} \cup G_e \cup G_{v1}\},\\
\mathcal{C}_{col1} &= \{\{b\ b\ b\} \Rightarrow G_{col1}\},\\
\mathcal{C}_0 &= \{\{X\ v\ X\} \Rightarrow \{X\ a\ Y\}\},\\
\mathcal{C}_1 &= \{\{X\ a\ Y\,.\,X\ a\ Z\,.\,Y\ e\ Z\} \Rightarrow \{b\ b\ b\}\}.
\end{aligned}$$

Note that $\mathcal{C}_{basic}$ and $\mathcal{C}_{col1}$ are not written as b-bounded sets in order to increase their readability. However, they could be easily by sets of b-bounded tgds. Finally, let $\mathcal{P}$ contain a single SPARQL graph pattern P defined as

$$\begin{aligned}
P = {} & (and(\{C_i\ iscol\ C_i \mid v_i \in V_2\} \cup \{b\ b\ b\}))\ \text{UNION}\\
& (and(\{C_i\ iscol\ C_i \mid v_i \in V\}))\ \text{UNION}\\
& (and(\{C_i\ iscol\ C_i \mid v_i \in V_2\})\ \text{OPT}\\
& \quad and(\{C_i\ iscol\ C_i \mid v_i \in V_1\} \cup\\
& \qquad \{C_i\ e\ C_j \mid (v_i,v_j) \in E\}))
\end{aligned}$$

where $and(T)$ for a set $T = \{t_1,\dots,t_n\}$ of triple patterns $t_1,\dots,t_n$ denotes the graph pattern t_1 AND ... AND t_n, and a new variable C_i is introduced for each $v_i \in V$. The reduction is obviously feasible in LOGSPACE. A detailed correctness proof is omitted, but a few important properties are noted:

- Obviously, $[\![P]\!]_G$ contains exactly all possible mappings $\mu\colon \{C_i \mid v_i \in V_2\} \to \{0,1,2\}$ (because of the part before the first UNION), and all possible mappings $\mu\colon \{C_i \mid v_i \in V\} \to \{0,1,2\}$ (because of the middle part).
- Every $G' \subset G$ that satisfies $\mathcal{C}$ contains $G_{cols} \cup G_e \cup G_{v1}$, and for each $v_i \in V_1$ exactly one triple from G_{col1}. Further, it does not contain $b\ b\ b$.
- Hence $[\![P]\!]_{G'}$ still contains all mappings $\mu\colon \{C_i \mid v_i \in V\} \to \{0,1,2\}$ due to the second UNION part. However, the first part does not contribute any mappings over G'.
- It can be easily checked that therefore $[\![P]\!]_{G'} = [\![P]\!]_G$ if the coloring on V_1 encoded by those triples from G_{col1} that are contained in G' cannot be extended to a valid 3-coloring on V: In this case, every possible mapping $\mu\colon \{C_i \mid v_i \in V_2\} \to \{0,1,2\}$ cannot be extended to a mapping that also maps the OPT part of the last part of the query.

□

A complete and more fine-grained analysis of different fragments of SPARQL is left to future work.

6. Problem variations

The goal of this section is twofold. First, a variation of the graph minimisation problem is considered that might be able to give more information about the amount of redundancy in an RDF graph than the problems considered so far. Second, note that throughout the paper the goal was not only to just classify the problems according to their computational complexity, but also to identify the reasons and sources of their complexity. The remaining problems studied in this section are thus thought to further pinpoint the source of complexity of some of the settings studied previously.

6.1. Minimisation under size bounds

For the graph minimisation problems, the question considered so far was if an RDF graph G contains at least one redundant triple, with respect to the given set of rules, constraints, and possibly queries. However, this does not give much information about the amount of redundancy in the data. An alternative question of interest is thus if a graph can be reduced below some predefined size, without losing any information. This question is formalised as follows.

Definition 6.1. *Let* MINI-RDF$^{card}(G, \mathcal{R}, \mathcal{C}, k)$ *be the following decision problem:*
INPUT: An RDF graph G, a set $\mathcal{R}$ of RDF rules, a set $\mathcal{C}$ of tgds (s.t. G satisfies $\mathcal{C}$) and an integer k.
QUESTION: Does there exist a subgraph $G' \subset G$ with $|G'| \leq k$, s.t. G' satisfies $\mathcal{C}$ and $G \subseteq Cl_{\mathcal{R}}(G')$?

It can be easily verified that for all settings in Table 1 that are at least NP-hard, the complexity of MINI-RDF^{card} is the same as for MINI-RDF$^{\subseteq}$. Intuitively, this is because the nondeterministic algorithms (resp. the deterministic algorithms using NP oracles) for solving these problems at some point all contain a statement

– *Guess a subgraph $G' \subset G$.*

Replacing this statement by

– *Guess a subgraph $G' \subset G$ with $|G'| \leq k$*

immediately solves MINI-RDF^{card}. Therefore, the only two interesting cases remaining are MINI-RDF$^{\subseteq}$ with a b-bounded or fixed set $\mathcal{R}$ and no constraints, as they can be decided in PTIME. The next theorem shows that the tractability for deciding if G contains at least one redundant triple does not carry over to checking if there is a certain amount of triples redundant. Intuitively, the reason for this is that it is no longer sufficient to check a polynomial number of subgraphs.

Theorem 6.1. *The problem* MINI-RDF$^{card}(G, \mathcal{R}, \mathcal{C}, k)$ *is* NP*-complete if $\mathcal{C} = \emptyset$ and $\mathcal{R}$ is either considered as fixed or a set of b-bounded rules (for fixed b).*

Proof. Membership carries over from the NP membership of MINI-RDF$^{card}(G, \mathcal{R}, \mathcal{C}, k)$ discussed above where $\mathcal{C}$ is a set of b-bounded tgds and $\mathcal{R}$ is a set of arbitrary rules.

Hardness is shown by reduction from the well-known NP-complete problem *Vertex Cover* on graphs.

Hence consider a fixed set $\mathcal{R}$ of rules:

$$\begin{aligned}\mathcal{R} = \{ \; & \{V \; neighbour \; E \, . \, V \; v \; V\} \Rightarrow \{E \; e \; E\}; \\ & \{E \; e \; E \, . \, X \; in \; X \, . \, X \; succ \; E\} \Rightarrow \{E \; in \; E\}; \\ & \{E \; last \; E \, . \, E \; in \; E \, . \, V \; backupv \; V\} \\ & \qquad \Rightarrow \{V \; v \; V\} \; \}.\end{aligned}$$

Now let an arbitrary instance of Vertex Cover be given by a graph $G = (V, E)$ and an integer k', where $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$. Without loss of generality, assume G to contain no self-loops, this means edges (v_i, v_i). Then define an instance of MINI-RDF^{card} $(G^{rdf}, \mathcal{R}, \mathcal{C}, k)$ as follows: $\mathcal{R}$ is as defined above and $\mathcal{C} = \emptyset$. For defining G, assume an arbitrary order on the edges in E (i.e. let $E = (e_1, \ldots, e_m)$). Then $G^{rdf} = G_v \cup G_{nghb} \cup G_{ord}$, where

$$\begin{aligned}G_{nghb} &= \{v_i \; neighbour \; e_\ell \mid v_i \in V, e_\ell \in E, \\ &\qquad\qquad e_\ell = (v_i, v_j)\}, \\ G_{ord} &= \{e_i \; succ \; e_{i+1} \mid i \in \{1, \ldots, m-1\}\} \cup \\ &\quad \{e_m \; last \; e_m \, . \, e_0 \; in \; e_0 \, . \, e_0 \; succ \; e_1\}, \text{ and} \\ G_v &= \{v_i \; v \; v_i \, . \, v_i \; backupv \; v_i \mid v_i \in V\}.\end{aligned}$$

Thereby introduce a new URI v_i for ever $v_i \in V$, a new URI e_ℓ for every $e_\ell \in E$ (by slight abuse of notation, let v_i and e_ℓ denote both, the vertex or edge in the input graph and the URI in G). Note that G_{ord} is defined according to the arbitrary order assumed on E and e_0 is some fresh URI, representing a "dummy" edge. Finally, set $k = 3 * m + 2 + k' + |V|$.

The reduction is obviously feasible in LOGSPACE. Before showing that it is indeed correct, its intuition is sketched. Note that the only triples from G^{rdf} that can be derived by $\mathcal{R}$ are such of the from $v_i \; v \; v_i$. Therefore the idea is that those triples with v on predicate position remaining in some valid subgraph G' encode a valid vertex cover. To ensure this, the first rule in $\mathcal{R}$ adds a triple $e_j \; e \; e_j$ for every edge covered by G'. The second rule adds $e_j \; in \; e_j$ to G' if all predecessors of e_j according to the assumed arbitrary order are covered by G'. Hence if $e_m \; in \; e_m$ can be derived for the last edge e_m in this order, then all edges are indeed covered. If this is the case, the last rule allows us to re-insert again the triples $v_i \; v \; v_i$ for all vertices not being part of the vertex cover, hence the complete graph G^{rdf}.

The proof of the correctness of this reduction can be found in the appendix. □

6.2. Minimising without rules

In the previous sections, for the settings where $\mathcal{C}$ may contain arbitrary tgds, the complexity of MINI-RDF$^{\models}$ and MINI-RDF$^{\subseteq}$ was significantly higher than for all other settings. Hence a natural question is whether this higher complexity is due to the missing restrictions on $\mathcal{C}$ only, or whether it arises from the interplay of all components of the setting. The next theorem gives an answer to this question by showing that already the question whether there exists some non-empty subgraph that satisfies all constraints contains the full hardness.

Theorem 6.2. *Let G be an RDF graph and $\mathcal{C}$ a set of tgds. Deciding whether there exists some subgraph $G' \subset G$ ($G' \neq \emptyset$) such that G' satisfies $\mathcal{C}$ is Σ_3^P-complete.*

The proof, which is provided in the appendix, is an appropriate adaption of the proof of Lemma 3.5. Recall that there certain triples must not be removed from G since no rule was provided in order to recover them. In contrast, all these triples are now explicitly required to remain in any "valid" subgraph of G. This is done by tgds of the form $\{S\ P\ O\} \Rightarrow \{t\}$ that are satisfied over any nonempty RDF graph iff t is contained in this graph.

Next, recall that tgds generalise (safe) Datalog rules by allowing existential quantification and conjunctions in the head. In other words, Datalog rules are an important special case of tgds – referred to as *full tgds* in the information integration literature. While being less expressive than tgds, many reasoning tasks become easier (or decidable) in the presence of full tgds, compared to arbitrary tgds. The next theorem shows that this holds also true for the problems considered in this paper. Restricting the constraints to full tgds pushes the Σ_3^P-completeness results from Theorems 3.1 and Theorem 6.2 down to Σ_2^P.

Theorem 6.3. *The problems* MINI-RDF$^{\models}(G, \mathcal{R}, \mathcal{C})$ *and* MINI-RDF$^{\subseteq}(G, \mathcal{R}, \mathcal{C})$ *are Σ_2^P-complete if $\mathcal{C}$ is a set of full tgds and $\mathcal{R}$ is a set of arbitrary rules. The problem remains Σ_2^P-complete even if $\mathcal{R}$ is fixed.*

Likewise, let G be an RDF graph and $\mathcal{C}$ a set of full tgds. Deciding whether there exists some subgraph $G' \subset G$ with $G' \neq \emptyset$ such that G' satisfies $\mathcal{C}$ is Σ_2^P-complete.

Recall from the Σ_3^P-hardness proofs from Lemma 3.5 and Theorem 6.2 how the quantifier alternation was encoded. The first existential quantifier block was encoded in the selection of the subgraph. The following block of universal quantifiers was encoded in the antecedent of one big tgd, and the last block of existential quantifier was encoded on its consequent. The reduction remains basically the same, only that because of the missing existential quantifier in the consequent of full tgds, the last quantifier block can no longer be encoded. Hence instead a reduction from Q-3COL$_{\exists,3}$, only a reduction from Q-3COL$_{\exists,2}$ is possible.

To conclude this subsection, note that just like Theorem 6.2, a similar point can also be made for the NP-complete settings in Theorem 3.1. From the proof of Lemma 3.7, it already follows that for MINI-RDF$^{\models}$, one source of the NP-hardness is just to decide entailment. However, there exists yet another source for the NP-hardness of the settings allowing for at least b-bounded tgds. In those cases, already testing for the existence of some subgraph that satisfies all constraints is NP-hard. This is formalised in the next theorem.

Theorem 6.4. *Let G be an RDF graph and $\mathcal{C}$ a set of b-bounded tgds. Deciding whether there exists some subgraph $G' \subset G$ such that $G' \neq \emptyset$ and G' satisfies $\mathcal{C}$ is* NP*-complete.*

Again, this theorem can be proved by a similar reduction as Lemma 3.7. Basically, all that needs to be changed is instead of not providing rules to derive triples that must not be removed from the RDF graph to explicitly enforce them via tgds. The concrete reduction is given in the appendix.

Note that as a result, the set $\mathcal{C}$ is no longer fixed (as it was in the reduction presented in the proof of Lemma 3.7), but depends on (V, E).

6.3. General RDF rules vs. Datalog rules

This section is concluded by showing that the complexity of the investigated problems remains unchanged by allowing additional predicates $uri(.)$, $blank(.)$, $lit(.)$ to restrict the type of a value in a Datalog rule, that is, allowing general RDF rules as defined in Section 2. Note that for every $x \in U \cup B \cup L$ occurring in some RDF-graph G (i.e. for every element of the active domain) it can be easily recognised whether it belongs to U, B or L: This could be either decided using syntactic criteria, or by a lookup in U, B and L (although those sets are supposed to be countably infinite, one can assume that U_G, B_G and L_G, i.e. the elements of the active domain, are the "first" elements of these sets). Therefore, determining the type of some element requires at most polynomial time in the size of G. Therefore, for every element x of the active domain of G, we create a ground atom $B_t(x)$, $U_t(x)$ or $L_t(x)$, depending on the type of x. By encoding an atom $blank(X)$ as triple $\{X\ blank\ X\}$ in G, we can make this information available for rule application without increasing the complexity of the problem.

The same argument allows us to overcome the problem that the closure with respect to a rule set $\mathcal{R}$ contains invalid RDF triples (containing e.g. a blank node in a predicate position). Depending on whether invalid triples are allowed in intermediate results or not, we can pursue one of the following two strategies: (i) in a post-processing step, we can check for every triple in $\mathcal{R}(G)$ whether it is valid or not. In the latter case, it is removed; (ii) if invalid triples should also be excluded from any intermediate results, then the rules can be (automatically) augmented by at most 2 additional predicates in the rule body, $urib(A)$ and $uri(B)$, assuming that the rule head is $\{A\ B\ C\}$. The predicate $urib(.)$ can be easily defined from $uri(.)$ and $blank(.)$ by e.g. $\{uri(X)\} \Rightarrow \{urib(X)\}$ and $\{blank(X)\} \Rightarrow \{urib(X)\}$. This is similar to variations of rules (2)–(4) in Section 1, where the filter conditions guaranteed valid intermediate triples.

7. Conclusion and future work

In this paper, a collection of complexity results for minimisation problems over RDF graphs was proved, considering various restrictions on the rules and tgds. One such restriction was b-boundedness [19]. Note that this restriction can be relaxed by bounding not necessarily the size of the rules (or tgds) but only the maximal number of blank nodes occurring in the rules (or tgds) — in the Datalog world, Vardi [30] showed that such a restriction decreases complexity. Further, it was discussed how the complexity of the problem increases if one requires completeness only with respect to a given set of conjunctive queries (CQs). Notably, if the CQs are restricted to have bounded head arity, while providing additional minimisation potential, the problem becomes only mildly harder.

The minimisation problems considered here are driven by practical needs to represent RDF data compactly or tailor them to engines supporting different rule sets. The results also provide a basis for eliminating redundancies in existing practically relevant rule sets, such as OWL2RL [20]. We believe that our results will gain even more relevance with the advent of novel standards such as the W3C rule interchange format (RIF) which will allow one to enrich RDFS and OWL with Web-publishable custom rule sets [6].

Although in practice the interest lies rather on the construction problem (i.e. computing a redundancy free subgraph of a given RDF graph), this paper concentrates on the decision problem. The point of this work is to provide foundational research towards redundancy elimination in RDF graphs, aiming at a deeper understanding of the different sources of complexity of redundancy detection. For these tasks, the decision problem is the more appropriate problem to study. However, note that the complexity results for the decision problems naturally carry over to the corresponding construction problems. For example, the algorithms presented in the membership proofs all work by looking for a counter-example that witnesses the non-minimality of the current instance. Hence a naive algorithm for solving the construction problem would be to just iteratively apply the decision algorithms on these counter-examples until a minimal RDF graph is found.

As future work, our investigations should be further extended in several directions such as a more fine-grained analysis of SPARQL fragments when redundancy with respect to queries is considered. A first step towards this direction was done by proving first bounds for the fragment of well-designed SPARQL queries [24]. As a further step of future work, the investigation of new query features in the upcoming SPARQL1.1 version [13] such as aggregates, path queries, and subqueries is on our agenda. It also should be noted that our current observations based on Datalog apply a set-based semantics, whereas the SPARQL specification applies a bag (multiset) semantics: as another direction of future work it would be interesting to investigate redundancies under the point of view whether they affect duplicate solutions under SPARQL's bag semantics.

Last, but not least, we plan to cast the obtained results into practical algorithms to "compress" RDF graphs and rule sets, investigate related relevant problems such as "trading" triples for rules, or vice versa, and experimentally evaluating effects of such transformations on query answering with dynamic inference such as sketched in [17].

Acknowledgements

The research was funded by the Austrian Science Fund (FWF): P20704-N18, by the Vienna Science and Technology Fund (WWTF): ICT08-032, and by Science foundation Ireland (SFI) under grant SFI/08/-CE/I1380 (Líon-2). We thank the anonymous reviewers of the conference version and the reviewers of this paper for their helpful comments.

References

[1] M. Ajtai, R. Fagin, and L. J. Stockmeyer. The closure of monadic NP. *Journal of Computer and System Sciences*, 60(3):660–716, 2000.

[2] R. Angles and C. Gutierrez. The expressive power of SPARQL. In *The Semantic Web - ISWC 2008: 7th International Semantic Web Conference*, volume 5318 of *Lecture Notes in Computer Science*, pages 114–129. Springer, 2008.

[3] D. Beckett and T. Berners-Lee. Turtle - terse RDF triple language. W3C Team Submission, W3C, Jan. 2008. Available at `http://www.w3.org/TeamSubmission/turtle/`.

[4] C. Beeri and M. Y. Vardi. A proof procedure for data dependencies. *Journal of the ACM*, 31(4):718–741, 1984.

[5] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia - a crystallization point for the web of data. *Journal of Web Semantics*, 7(3):154–165, 2009.

[6] J. de Bruijn. RIF RDF and OWL Compatibility. W3C Proposed Recommendation, W3C, June 2010. Available at `http://www.w3.org/TR/rif-rdf-owl/`.

[7] J. de Bruijn, A. Polleres, R. Lara, and D. Fensel. OWL$^-$. WSML Working Draft d20.1v0.2, WSML, May 2005. Final draft, Available at `http://www.wsmo.org/TR/d20/d20.1/v0.2/`.

[8] T. Eiter, W. Faber, M. Fink, and S. Woltran. Complexity results for answer set programming with bounded predicate arities and implications. *Annals of Mathematics and Artificial Intelligence*, 51(2-4):123–165, 2007.

[9] T. Eiter, M. Fink, H. Tompits, P. Traxler, and S. Woltran. Replacements in non-ground answer-set programming. In *Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 340–351. AAAI, 2006.

[10] G. Gottlob and P. Senellart. Schema mapping discovery from data instances. *Journal of the ACM*, 57(2), 2010.

[11] B. N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logics. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 48–57, 2003.

[12] C. Gutierrez, C. A. Hurtado, A. O. Mendelzon, and J. Pérez. Foundations of semantic web databases. *Journal of Computer and System Sciences*, 77(3):520–541, 2011.

[13] S. Harris and A. Seaborne. SPARQL 1.1 Query Language. W3C Working Draft, W3C, Jan. 2012. Available at `http://www.w3.org/TR/sparql11-query/`.

[14] P. Hayes. RDF semantics. W3C Recommendation, W3C, Feb. 2004. Available at `http://www.w3.org/TR/rdf-mt/`.

[15] A. Hogan and S. Decker. On the ostensibly silent 'W' in OWL 2 RL. In *Web Reasoning and Rule Systems - Third International Conference, RR 2009*, volume 5837 of *Lecture Notes in Computer Science*, pages 118–134. Springer, 2009.

[16] A. Hogan, A. Harth, and A. Polleres. Scalable authoritative owl reasoning for the web. *International Journal on Semantic Web and Information Systems*, 5(2):49–90, 2009.

[17] G. Ianni, T. Krennwallner, A. Martello, and A. Polleres. Dynamic querying of mass-storage RDF data with rule-based entailment regimes. In *The Semantic Web - ISWC 2009: 8th International Semantic Web Conference*, volume 5823 of *Lecture Notes in Computer Science*, pages 310–327. Springer, 2009.

[18] G. Lausen, M. Meier, and M. Schmidt. SPARQLing constraints for RDF. In *EDBT '08: Proceedings of the 11th international conference on Extending database technology: Advances in database technology*, pages 499–509. ACM, 2008.

[19] M. Meier. Towards Rule-Based Minimization of RDF Graphs under Constraints. In *Web Reasoning and Rule Systems - Second International Conference, RR 2008*, volume 5341 of *Lecture Notes in Computer Science*, pages 89–103. Springer, 2008.

[20] B. Motik, B. C. Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz. OWL 2 Web ontology language profiles. W3C Recommendation, W3C, Oct. 2009. Available at `http://www.w3.org/TR/owl2-profiles/`.

[21] B. Motik, I. Horrocks, and U. Sattler. Bridging the gap between owl and relational databases. In *Proceedings of the Sixteenth International World Wide Web Conference (WWW2007)*, pages 807–816. ACM, 2007.

[22] S. Muñoz, J. Pérez, and C. Gutiérrez. Minimal deductive systems for RDF. In *The Semantic Web: Research and Applications, 4th European Semantic Web Conference, ESWC 2007*, volume 4519 of *Lecture Notes in Computer Science*, pages 53–67. Springer, 2007.

[23] C. H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

[24] J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of SPARQL. *ACM Transactions on Database Systems*, 34(3), 2009.

[25] A. Pettorossi and M. Proietti. Transformation of logic programs. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 5, pages 697–787. Oxford University Press, 1998.

[26] R. Pichler, A. Polleres, S. Skritek, and S. Woltran. Redundancy elimination on RDF graphs in the presence of rules, constraints, and queries. In *Web Reasoning and Rule Systems - Fourth International Conference, RR 2010*, volume 6333 of *Lecture Notes in Computer Science*, pages 133–148. Springer, 2010.

[27] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. W3C Recommendation, W3C, Jan. 2008. Available at `http://www.w3.org/TR/rdf-sparql-query/`.

[28] H. J. ter Horst. Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. *Journal of Web Semantics*, 3(2-3):79–115, 2005.

[29] J. D. Ullman. *Principles of Database and Knowledge Base Systems*. Computer Science Press, New York, NY, USA, 1989.

[30] M. Vardi. On the complexity of bounded-variable queries. In *Proceedings of the Fourteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 266–276. ACM, 1995.

[31] K. W. Wagner. Bounded query classes. *SIAM Journal on Computing*, 19(5):833–846, 1990.

Appendix

A. Full proofs of Section 3

A.1. Proof of Lemma 3.5

Recall the reduction presented in Section 3. It remains to show that it indeed satisfies the given intuition, i.e. that $\mathcal{G}$ is a positive instance of Q-3COL$_{\exists,3}$ iff there exists $G' \subset G$ s.t. $Cl_{\mathcal{R}}(G) = Cl_{\mathcal{R}}(G')$ and G' satisfies $\mathcal{C}$. Before showing the both directions separately, in order to prove that the reduction indeed defines a valid instance of MINI-RDF$^{\subseteq}$, it is first shown that G satisfies $\mathcal{C}$: This trivially holds for $\mathcal{C}_{col1}$, and can also easily be checked for $\mathcal{C}_0$. Finally, for $\mathcal{C}_{\mathcal{G}}$, note the following: every possible mapping $\mu\colon \{C_i \mid v_i \in V_1 \cup V_2\} \to \{0,1,2\}$ is a homomorphism from $G_1 \cup G_2$ into G. However, every possible extension μ' of μ to $\{C_i \mid v_i \in V_3\}$ with $\mu'\colon \{C_i \mid v_i \in V\} \to \{0,1,2\}$ is obviously a homomorphism $G_3 \to G$. However, since G contains a triple with e on predicate position and every possible combination of $0,1,2$ on subject and object position, also $\mu'(G_4) \subset G$, which proves the case. In the following, by slight abuse of notation, the values $\{0,1,2\}$ of colorings for nodes $v_i \in V_1$ are identified with the URIs $\{0,1,2\}$ in G.

"if"-direction) Assume that there exists a subgraph $G' \subset G$ such that (1) G' satisfies $\mathcal{C}$ and (2) $Cl_{\mathcal{R}}(G) = Cl_{\mathcal{R}}(G')$ (or, equivalently, $G \subseteq Cl_{\mathcal{R}}(G')$). Then it must be shown that $\mathcal{G}$ is a positive instance of Q-3COL$_{\exists,3}$, i.e. that there exists a coloring σ_1 on V_1 such that for all colorings on V_2 there exists a coloring σ_3 on V_3 such that $\sigma_1 \cup \sigma_2 \cup \sigma_3$ is a valid 3-coloring of (V, E).

It is convenient to start with the following observations on G'

(i) $G_{cols} \cup G_{v1} \cup G_{col2} \cup G_{e2} \cup G_{neq} \subseteq G'$
(ii) $G_{e1} \cap G' = \emptyset$
(iii) For $v_i \in V_1$, let $G^i_{col1} = \{v_i\ a\ 0 \,.\, v_i\ a\ 1 \,.\, v_i\ a\ 2\} \subseteq G_{col1}$. Then $|G^i_{col1} \cap G'| = 1$ for every $v_i \in V_1$.

Property (i) follows immediately from the observation that $\mathcal{R}$ does not allow to derive triples with $iscol$, v, c, or neq on predicate position (hence $G_{cols} \cup G_{v1} \cup G_{col2} \cup G_{neq} \subseteq Cl_{\mathcal{R}}(G')$ iff they are already contained in G') and further $\mathcal{R}$ does not allow to derive triples with e on predicate position and different values on subject and object position (hence $G_{e2} \subseteq Cl_{\mathcal{R}}(G')$ iff $G_{e2} \subseteq G'$).
To see that also property (ii) holds, assume to the contrary that this is not the case, that is assume for some $t \in G_{e1}$ also $t \in G'$. Then, since G' satisfies $\mathcal{C}$ and because of property (i), in order to satisfy the first tgd in $\mathcal{C}_0$, it must be the case that $G_{e1} \subseteq G'$. Further, again because of property (i), in order to satisfy the second tgd in $\mathcal{C}_0$ it is easy to check that also $G_{col1} \subseteq G'$. Thus in order to satisfy $\mathcal{C}$ it must be the case that $G' = G$, which contradicts $G' \subset G$.
Property (iii) holds because of two reasons: First of all, for every $v_i \in V_1$ it must be the case that $G^i_{col1} \cap G' \neq \emptyset$: It is easy to see that at least one triple in G^i_{col1} is needed to derive the two others by applying $\mathcal{R}$ (or $\mathcal{R}_{col1}$, to be precise). Hence $G^i_{col1} \cap G' = \emptyset$ would give a contradiction to G' satisfying property (2). On the other hand, if $|G^i_{col1} \cap G'| > 1$, then because of property (i) there exists a homomorphism from the antecedent of the only tgd in $\mathcal{C}_{col1}$ to G'. Hence in order to satisfy $\mathcal{C}_0$, G' must contain the triple $0\ e\ 0$. However, this contradicts property (ii).

With these additional three properties at hand, given G', define a 3-coloring σ_1 on V_1 for every $v_i \in V_1$ as $\sigma_1(v_1) = \alpha$ if the triple $v_i\ a\ \alpha \in G'$. By property (iii), this 3-coloring is well defined. It remains to show that σ_1 is indeed the desired 3-coloring. Towards this goal, consider an arbitrary 3-coloring σ_2 on V_2, and define a mapping $\mu\colon \{C_i \mid v_i \in V_1 \cup V_2\} \to \{0,1,2\}$ as follows: Let $\sigma' = \sigma_1 \cup \sigma_2$. Then $\mu(C_i) = \sigma'(v_i)$. Now it is easy to see that $\mu(G_1 \cup G_2) \subseteq G'$: $\mu(G_1) \subseteq G'$ follows from the definition of σ_1, and $\mu(G_2) \subseteq G'$ holds because for every possible coloring $\alpha = \sigma_2(v_i)$ for $v_i \in V_2$, the graph $G_{col2} \subseteq G'$ (Property (i)) contains a triple $v_i\ c\ \alpha$. Hence, since G' satisfies $\mathcal{C}_{\mathcal{G}}$ by assumption, there must exist an extension μ' of μ to $\{C_i \mid v_i \in V_3\}$ s.t. $\mu'(G_3 \cup G_4) \subseteq G'$. From μ', define the following 3-coloring σ_3 on V_3: for $v_i \in V_3$, let $\sigma_3(v_i) = \mu'(C_i)$. Since it is easy to see that $\mu(G_3) \subseteq G_{col2}$, it follows that σ_3 is indeed well defined. Hence it remains to show that $\sigma = \sigma_1 \cup \sigma_2 \cup \sigma_3$ is a valid 3-coloring for (V, E). This follows immediately from Property (ii), the definition of G_4 and the observation that $\mu'(C_i) = \sigma(v_i)$ for all $v_i \in V$: By definition, G_4 contains one triple for every edge in E. Since by Property (ii) G' contains no triple with e on predicate position and the same value on subject and predicate position, it must be the case $\mu'(v_i) \neq \mu'(v_j)$ for every triple $v_i\ e\ v_j \in G_4$. Hence for every edge $(v_i, v_j) \in E$ it holds that $\sigma(v_i) \neq \sigma(v_j)$, which proves the case.

"only if"-direction) Assume that $\mathcal{G}$ is a positive instance of Q-3COL$_{\exists,3}$, i.e. that there exists a coloring σ_1 on V_1 such that for all colorings on V_2 there exists a coloring σ_3 on V_3 such that $\sigma_1 \cup \sigma_2 \cup \sigma_3$ is a valid 3-coloring of (V, E). Then it must be shown that there exists a subgraph $G' \subset G$ such that (1) G' satisfies $\mathcal{C}$ and (2) $Cl_{\mathcal{R}}(G) = Cl_{\mathcal{R}}(G')$ (or, equivalently, $G \subseteq Cl_{\mathcal{R}}(G')$).

Towards this goal, define $G' \subset G$ as follows. Let $G'_{col1} = \{v_i\ a\ \alpha \mid v_i \in V_1, \alpha = \sigma_1(v_i)\}$. Then $G' = G_{cols} \cup G_{v1} \cup G_{col2} \cup G_{e2} \cup G_{neq} \cup G'_{col1}$.

Now it is easy to see that G' satisfies property (2): Since obviously $G \setminus G' = G_{e1} \cup (G_{col1} \setminus G'_{col1})$, it suffices to show that $G_{col1} \subseteq Cl_{\mathcal{R}}(G')$ and $G_{e1} \subseteq Cl_{\mathcal{R}}(G')$. Now $G_{e1} \subseteq Cl_{\mathcal{R}}(G')$ because of $\mathcal{R}_{e1}$ and $G_{cols} \subset G'$. To see that also $G_{col1} \subset Cl_{\mathcal{R}}(G')$, first note that G'_{col1} contains exactly one triple $v_i\ a\ \sigma_1(v_i)$ for every $v_i \in V_1$, since σ_1 assigns a color to every $v_i \in V_1$. Hence, since also $G_{cols} \subset G'$, it is easy to check that this allows to derive G_{col1} from G' via $\mathcal{R}_{col1}$.

It remains to show that G' also satisfies $\mathcal{C}$.
For the tgd in $\mathcal{C}_{col1}$, this is easy to see since by definition of G'_{col1}, for every $v_i \in V_1$ the graph G' contains exactly one triple $v_i\ a\ \alpha$ where $\alpha \in \{0,1,2\}$. Hence the antecedent of the tgd cannot be mapped into G'.
For the tgds in $\mathcal{C}_0$, it is again easy to see that the antecedent cannot be mapped into G': The only triples $t \in G$ with e on predicate position that have the same value on subject and object position are the triples in G_{e1}. Since $G_{e1} \cap G' = \emptyset$, the triple $X\ e\ X$ cannot be mapped into G'.
Finally consider the tgd $\forall\{C_i \mid v_i \in V_1 \cup V_2\} G_1 \cup G_2 \Rightarrow \exists\{C_i \mid v_i \in V_3\} G_3 \cup G_4$, and let μ be an arbitrary homomorphism from $G_1 \cup G_2$ into G'. Then the following claim holds on the relationship between σ_1 and μ: For all $v_i \in V_1, \sigma_1(v_i) = \alpha$ iff $\mu(C_i) = \alpha$. This is due to the construction of G', since G'_{col1} contains exactly that triple $t_i \in \{v_i\ a\ 0\,.\,v_i\ a\ 1\,.\,v_i\ a\ 2\}$ with $t_i = v_i\ a\ \alpha$. Hence μ must map the triple $v_i\ a\ C_i$ onto t_i, and therefore $\mu(C_i) = \alpha$.

Next define a coloring σ_2 on V_2 from μ as follows: For every $v_i \in V_2$, let $\sigma_2(v_i) = \alpha$ where $\alpha = \mu(C_i)$. Since this obviously gives a coloring on V_2 (every $v_i \in V_2$ is assigned exactly one "color" from $\{0,1,2\}$), by assumption there must exist a coloring σ_3 on V_3 such that $\sigma_1 \cup \sigma_2 \cup \sigma_3$ is a valid 3-coloring on (V,E). From σ_3, define the following extension μ' of μ to $\{C_i \mid v_i \in V_3\}$: $\mu'(C_i) = \sigma_3(v_i)$. Obviously, μ' is well-defined. Now the final claim is that $\mu'(G_3 \cup G_4) \subseteq G'$. To see this, first consider $\mu'(G_3)$: Since for every $v_i \in V_3$ we have $\sigma_3(v_i) \in \{0,1,2\}$, also $v_i\ c\ \mu'(C_i) \in \{v_i\ c\ 0\,.\,v_i\ c\ 1\,.\,v_i\ c\ 2\} \subseteq G_{col2} \subseteq G'$. Hence $\mu'(G_3) \subseteq G'$. It therefore only remains to show that also $\mu'(G_4) \subseteq G'$. To see this, let $\sigma = \sigma_1 \cup \sigma_2 \cup \sigma_3$ and note the following relationship between μ' and σ: For all $v_i \in V$, $\mu'(C_i) = \sigma(v_i)$. This holds because of the corresponding relationship between μ and σ_1 and the fact that σ_2 was defined from μ and the extension of μ on $\{C_i \mid v_i \in V_3\}$ to μ' was defined based on σ_3. From this and the fact that σ is a valid 3-coloring for (V,E) (that is there does not exist an edge $(v_i, v_j) \in E$ such that $\sigma(v_i) = \sigma(v_j)$) it follows immediately that $\mu'(C_i)\ e\ \mu'(C_j) \in G_{e2}$, hence $\mu'(G_4) \subseteq G'$. This shows that G' also satisfies $\mathcal{C}_{\mathcal{G}}$.

Hence it was shown that G' satisfies (1) and (2), which concludes the proof.

So far the proof only showed hardness for MINI-RDF$^{\subseteq}$. However, note that G contains no blank nodes. As all rules in $\mathcal{R}$ are save, also $Cl_{\mathcal{R}}(G)$ cannot contain any blank node. Hence $Cl_{\mathcal{R}}(G') \models Cl_{\mathcal{R}}(G)$ iff $Cl_{\mathcal{R}}(G) \subseteq Cl_{\mathcal{R}}(G')$. Therefore the above proof also shows hardness for MINI-RDF$^{\models}$.

A.2. Proof of Lemma 3.7

It only remains to show the correctness of the reduction. First of all, it can be easily checked that G satisfies $\mathcal{C}$. The first tgd is trivially satisfied since $0\ neq\ 0 \in G$. For the second and third tgd just note that for all possible values $\mu(X)$ and $\mu(F)$ to which X and F (resp. $\mu(C)$ and $\mu(D)$ for C and D) can be mapped by a homomorphism μ from the antecedent of the tgd to G, there exists a the corresponding triple $\mu(X)\ a\ \mu(F)$ (resp. $\mu(C)\ neq\ \mu(D)$) in G.

Next it must be shown that there exists a $G' \subset G$ s.t. (1) G' satisfies $\mathcal{C}$ and (2) $Cl_{\mathcal{R}}(G) = Cl_{\mathcal{R}}(G')$ (or, equivalently, $G \subseteq Cl_{\mathcal{R}}(G')$ iff $\hat{G}$ is a positive instance of 3COL. In the following, both directions are shown separately.

"if"-direction) Assume that $\hat{G}$ is a positive instance of 3COL, this means that there exists a valid 3-coloring σ on V. Then it must be shown that there exists a subgraph $G' \subset G$ that satisfies (1) and (2).

Towards this goal, define $G' \subset G$ as follows: Let $G'_v = \{v_i\ a\ \sigma(v_i) \mid v_i \in V\}$, and $G' = G_{cols} \cup G_e \cup G_{neq} \cup G'_v$. It remains to show that G' satisfies the required properties (1) and (2).

First of all, it is easy to see that it satisfies (2): Since $G_{cols} \subseteq G'$, the second rule in $\mathcal{R}$ allows to derive G_{eq}. Further, note that by definition for every $v_i \in V$ the graph G'_v contains exactly one triple with v_i at subject and a on predicate position. Hence, because of this and since $G_{cols} \subseteq G$, the first rule allows to derive G_v from G'. This shows that $G \subseteq Cl_{\mathcal{R}}(G')$, and therefore proves the case.

Hence it remains to show that G' satisfies $\mathcal{C}$.
Since $0\ neq\ 0 \notin G'$, in order to satisfy the first tgd, there must not be a homomorphism from the antecedent into G'. Now since $G_{eq} \cap G' = \emptyset$, every such homomorphism must map C and D to different values.

However, as already stated above, by definition for every $v_i \in V$ the graph G'_v contains exactly one triple with v_i at subject and a on predicate position. Hence such a homomorphism cannot exist.
For the second and third tgd, just note that since $G_{eq} \cap G' = \emptyset$ there cannot exist a homomorphism from neither of the two antecedents into G'.
For the last tgd, assume an arbitrary homomorphism μ from its antecedent into G'. Then $(v_i, v_j) \in E$ holds for $v_i = \mu(X)$ and $v_j = \mu(Y)$ by the definition of G_e and the fact that μ must map the triple $X\ e\ Y$ into some triple of G_e. Further, by definition of G'_v, it must be the case that $\mu(C) = \sigma(v_i)$ and $\mu(D) = \sigma(v_j)$. Hence $\mu(D), \mu(C) \in \{0, 1, 2\}$ and $\mu(D) \neq \mu(C)$. Therefore the triple $\mu(C)\ neq\ \mu(D)$ is obviously contained in $G_{neq} \subseteq G'$. This shows that G' satisfies also the last tgd and therefore concludes the first direction of the proof.

"only if"-direction) Assume that there exists a subgraph $G' \subset G$ that (1) G' satisfies $\mathcal{C}$ and (2) $Cl_{\mathcal{R}}(G) = Cl_{\mathcal{R}}(G')$ (or, equivalently, $G \subseteq Cl_{\mathcal{R}}(G')$). Then it must be shown that $\hat{G}$ is a positive instance of 3COL, that is that there exists a valid 3-coloring σ on V.

For $v_i \in V$, let $G^i_v = \{v_i\ a\ 0\,.\, v_i\ a\ 1\,.\, v_i\ a\ 2\}$. It is convenient to start by observing the following properties of G':

(i) $G_{cols} \cup G_e \cup G_{neq} \subseteq G'$
(ii) $G^i_v \cap G' \neq \emptyset$ for every $v_i \in V$.
(iii) $G_{eq} \cap G' = \emptyset$
(iv) $|G^i_v \cap G'| = 1$ for every $v_i \in V$.

To see that Property (i) holds note that $G_{cols} \cup G_e \subseteq G'$ since $\mathcal{R}$ does not allow to derive triples with $iscol$ or e on predicate position. Further, $G_{neq} \subseteq G'$ since $\mathcal{R}$ does not allow to derive triples with neq on predicate position and two different values on subject and object position.
Property (ii) follows from the easy observation that if $G^i_v \cap G' = \emptyset$ for some $v_i \in V$, then $G^i_v \cap Cl_{\mathcal{R}}(G') = \emptyset$ as well, since the first rule requires at least one $t \in G^i_v$ to be contained in G' in order to derive G^i_v.
For property (iii), assume to the contrary that for some triple $t \in G_{eq}$ also $t \in G'$. Then because of $G_{cols} \subseteq G'$ (by Property (i)), also $G_{eq} \subseteq G'$ in order to satisfy the third tgd. Further, since $G_{col} \subseteq G'$ and because of Property (ii) also at least one triple from each G^i_v (for $v_i \in V$) is contained in G', then $G_v \subseteq G'$ in order to satisfy the second tgd. Hence $G \setminus G' = \emptyset$ gives the desired contradiction.
For Property (iv), $|G^i_v \cap G'| \geq 1$ follows from Property (ii). Now assume $|G^i_v \cap G'| \geq 2$. Then there obviously exists a homomorphism from the antecedent of the first tgd into G'. Hence in order to satisfy this constraint, $0\ neq\ 0 \in G'$ must hold, which contradicts Property (iii), and therefore proves the case.

Having shown these properties, let a 3-coloring σ of V on every $v_i \in V$ be defined as $\sigma(v_i) = \alpha$ if the triple $v_i\ a\ \alpha$ is in G'. Since by Property (iv) for every $v_i \in V$ there exists exactly one such triple in G', obviously σ is well defined. In order to see that σ is indeed a valid 3-coloring, consider an arbitrary edge $(v_i, v_j) \in E$. Define a mapping μ on the antecedent of the fourth tgd as follows: $\mu(X) = v_i$, $\mu(Y) = v_j$, $\mu(C) = \alpha$, and $\mu(D) = \beta$ where $\alpha = \sigma(v_i)$ and $\beta = \sigma(v_j)$. Now obviously, μ is a homomorphism from the antecedent of the fourth tgd into G': Beside the triples $v_i\ a\ \alpha$ and $v_j\ a\ \beta$, also $v_i\ e\ v_j$ is contained in G', since $v_i\ e\ v_j \in G_e \subseteq G'$. Since G' satisfies $\mathcal{C}$, G' contains also the triple $\mu(C)\ neq\ \mu(D) = \alpha\ neq\ \beta$. Hence because of Property (iii) it follows that $\alpha \neq \beta$ which proves the case and concludes the proof.

B. Full proofs of Section 4

B.1. Proof of Lemma 4.3

First of all, the reduction presented in Section 4 is obviously feasible in LOGSPACE. Next it remains to show the correctness of the reduction. The structure of this proof is as follows: First, the missing proof of Claim 1 is given. After this, the remaining correctness proof follows. In the following, denote with G_i the graph $G_i = \{X_r\ E\ X_s \mid 1 \leq r < s \leq i\}$, contained in the antecedent of each rule r_i.

Claim 1: Let $\hat{G}$ be an arbitrary instance of ODD CLIQUE and $(G, \mathcal{R})$ be defined as above. Then none of the rules $r_i \in \mathcal{R}$ for $i \in [n]$ is redundant.

Proof of Claim 1: To see that this claim is indeed true, just note that for every $i \in [n]$ the following mapping h is a homomorphism from the antecedent of r_i into G: $h(E) = e^*$, and $h(X_j) = v_j$ for $j \in [i]$. It can be easily checked that $h(\{E\ s\ E\}) \subseteq G_{eh}$ and $h(\{X_r\ E\ X_s\}) \subseteq G_{e2}$ for $1 \leq r < s \leq i$. Hence the triple $e^*\ o_i\ e^*$ respectively $e^*\ e_i\ e^*$ is contained in $Cl_{\mathcal{R}}(G)$ for every $i \in [n]$, depending of i being odd or even. Now it is easy to check that r_i is the only rule in $\mathcal{R}$ that can derive a triple with e^* on subject and object position, and o_i (resp. e_i) on predicate position. This concludes the proof.

Hence from the claim we know that if $\mathcal{R}$ contains some redundant rule, then it is the last one. Therefore the correctness of the proof comes down to showing that this rules is redundant iff the size of the biggest clique in the given graph is odd.

For the correctness proof, it is convenient to formulate a second claim:

Claim 2: For $i \in \{2, \dots, n\}$, there exists a homomorphism $h_i \colon G_i \to G_{e1}$ iff $\hat{G}$ contains a clique of size i.

Proof of Clam 2: First, assume that there exists a clique $C = \{v_{\alpha_1}, \dots, v_{\alpha_i}\}$ of size i. Then define h_i as $h_i(E) = e$ and $h_i(X_j) = v_{\alpha_j}$ for $j \in [i]$. Since C is a clique, $(v_{\alpha_r}, v_{\alpha_s}) \in E$ for all $1 \leq r < s \leq i$. From this it is easy to see that h_i is indeed a homomorphism. Next, assume that there exists a homomorphism $h_i \colon G_i \to G_{e1}$. Then obviously $h_i(E) = e$. Further, let $C \subseteq V$ be defined as $C = \{h_i(X_1), \dots, h_i(X_i)\}$ (where by slight abuse of notation the nodes $v_i \in V$ are identified with their corresponding URIs v_i). To see that C is a clique of size i, first note that for every pair (X_r, X_s) of variables with $1 \leq r < s \leq i$, there exists a triple $t = X_r\ E\ X_s$ in G_i. From the fact that $h_i(t) \in G_{e1}$, two observations follow: First, for every pair $(h_i(X_r), h_i(X_s)) \in C$, there exists an edge $(h_i(X_r), h_i(X_s)) \in E$, hence C is indeed a clique. Second, since E does not contain self loops, $h_i(X_s) \neq h_i(X_r)$ holds by the definition of G_{e1}. Hence C contains indeed i different nodes.

It remains to formally show the correctness of the reduction, i.e. to show that the size of the biggest clique in $\hat{G}$ is odd iff there exists a subset $\mathcal{R}' \subset \mathcal{R}$ such that $Cl_{\mathcal{R}}(G) = Cl_{\mathcal{R}'}(G)$.

It follows from Claim 1 that the only rule in $\mathcal{R}$ that may be redundant is $r = \{E\ c\ O\ .\ e\ E\ e\} \Rightarrow \{e\ O\ e\}$. Denote with $\mathcal{R}'$ the set $\mathcal{R}' = \mathcal{R} \setminus \{r\}$. Hence it remains to show that r is redundant iff the size of the biggest clique in $\hat{G}$ is odd. Towards this goal, notice that rule r is redundant iff for every triple $e\ e_i\ e$ in $Cl_{\mathcal{R}'}(G)$ s.t. there exists a triple $e_i\ c\ o_{i+1}$ in G, also the triple $e\ o_{i+1}\ e$ is in $Cl_{\mathcal{R}'}(G)$. Hence it suffices to show that this is the case iff the size of the biggest clique in $\hat{G}$ is odd.

To see that this holds, note that by Claim 2 and the definition of r_i, the triple $e\ e_i\ e$ is in $Cl_{\mathcal{R}'}(G)$ iff $\hat{G}$ contains a clique of size i (for i even). On the other hand, the triple $e\ o_i\ e$ is in $Cl_{\mathcal{R}'}(G)$ iff $\hat{G}$ contains a clique of size i (for i odd).

Next assume that the size of the biggest clique in $\hat{G}$ is odd (say m). Since $E \neq \emptyset$ it follows that $m \geq 3$. Further, it follows from the arguments above that $Cl_{\mathcal{R}'}(G)$ contains the triples $\{e\ o_m\ e\ .\ e\ o_{m-2}\ e\ .\ \dots\ .\ e\ o_3\ e\}$. On the other hand, $Cl_{\mathcal{R}'}(G)$ does not contain the triples $\{e\ e_{m+1}\ e\ .\ e\ e_{m+3}\ e\ .\ \dots\}$. Hence r is indeed redundant.

Finally assume that the size of the biggest clique in $\hat{G}$ is even (say $m \leq n$). Since $E \neq \emptyset$, it follows that $2 \leq m$. Hence the triple $e\ e_m\ e \in Cl_{\mathcal{R}'}(G)$. But since $\hat{G}$ does not contain a clique of size $m+1$, the triple $e\ o_{m+1}\ e$ is not in $Cl_{\mathcal{R}'}(G)$. However, since $m \leq n$ and m is an even number, the triple $e_m\ c\ o_{m+1}$ is in G, and therefore the triple $e\ o_{m+1}\ e$ can be derived from $Cl_{\mathcal{R}'}(G)$ via r. Therefore r is not redundant. This concludes the proof.

C. Full proofs of Section 5

C.1. Proof of Lemma 5.5

It remains to show that $\mathcal{G}$ is a positive instance of Q-3COL$_{3,\exists}$ iff $(G, \emptyset, \mathcal{C}, \mathcal{Q})$ is a positive instance of MINI-RDF$^{\subseteq, CQ}$. In the following, both directions are shown independently.

"If"-direction: Assume that there exists $G' \subseteq G$ that satisfies $\mathcal{C}$ and such that $q(G) = q(G')$. Then define a coloring σ_1 of V_1 as follows: For every $v_i \in V_1$, let $\{v_i\ c\ \alpha\} = G_v^i \cap G'$. Set $\sigma_1(v_i) = \alpha$ (by slight abuse of notation, the possible colors $\{0,1,2\}$ for V and the URIs $\{0,1,2\}$ in G are identified). Now let σ_2 be an arbitrary coloring of V_2. Then define a mapping μ on $C_{i_1}, \dots, C_{i_p}$ as $\mu(C_{i_\beta}) = \sigma(v_{i_\beta})$. Since by Claim 1 there exists at least one solution in $q(G')$ s.t. $C_2 = (\mu(C_{i_1}), \dots, \mu(C_{i_p}))$, there must exist a extension μ' of μ such that $\mu'(body(q)) \subseteq G'$. Now consider the coloring σ defined as $\sigma(v_i) = \mu'(C_i)$ for all $v_i \in V$. It is easy to see that σ is a valid 3-coloring on V: Consider an arbitrary edge $(v_i, v_j) \in E$. Then there exists a triple $C_i\ e\ C_j$ in $body(q)$. Therefore $\mu'(C_i) \neq \mu'(C_j)$ since $G_{e1} \cap G' = \emptyset$, hence $\sigma(v_i) \neq \sigma(v_j)$. Further note that it follows trivially that $\sigma(v_i) = \sigma_1(v_i)$ for all $v_i \in V_1$ and $\sigma(v_i) = \sigma_2(v_i)$ for all $v_i \in V_2$. Hence it was shown that there exists a coloring (σ_1) on V_1 such that for all possible colorings on V_2 there exists a coloring on V_3 such that the combined coloring gives a valid 3-coloring of (V, E), which proves the case.

"Only if"-direction: Assume that there exists a coloring σ_1 on V_1 such that for every coloring σ_2 on V_2 there exists a coloring σ_3 on V_3 such that $\sigma = \sigma_1 \cup \sigma_2 \cup \sigma_3$

is a valid coloring of (V, E). Then consider a subgraph $G' \subset G$ defined as $G' = G_{cols} \cup G_{v_1} \cup G_{e_2} \cup G_{neq} \cup G_c$ where $G_c = \{v_1\ c\ \sigma_1(v_i) \mid v_i \in V_1\}$ (note that again colorings are identified with the URIs $0, 1, 2$). Obviously, $G' \subset G$, hence it remains to show that G' satisfies $\mathcal{C}$ and that $q(G) = q(G')$.

It is easily checked that G' indeed satisfies $\mathcal{C}$: The first tgd is satisfied since its body cannot be mapped into G' (since by definition $G_{e1} \cap G' = \emptyset$). For the second tgd just note that since the value of $\sigma_1(v_i)$ is uniquely defined, $|G_v^i \cap G'| = 1$ for every $v_i \in V_1$, hence there cannot exists a homomorphism from the body of the second tgd into G' neither.

It remains to show that $q(G') = q(G)$, with $q(G)$ as specified in Claim 1. Hence consider an arbitrary tuple $(\mu(C), \mu(X), \mu(D_1), \mu(D_2), \mu(C_{i_1}), \dots, \mu(C_{i_p})) \in q(G)$, and consider an extension μ' of μ on $\{C_1, \dots, C_n\} \setminus \{C_{i_1}, \dots, C_{i_p}\}$ as follows: Let σ_1 be the coloring from above, and define a coloring σ_2 on V_2 as $\sigma_2(v_i) = \mu(C_i)$ for all $v_i \in V_2$. Now let σ_3 be a the coloring on V_3 such that $\sigma_1 \cup \sigma_2 \cup \sigma_3$ is a valid 3-coloring of (V, E). By assumption, σ_3 exists. Next define $\mu'(C_i) = \sigma_1(v_i)$ for all $v_i \in V_1$ and $\mu'(C_i) = \sigma_3(v_i)$ for all $v_i \in V_3$. It can now be easily checked that $\mu'(\mathit{body}(q)) \subseteq G'$. The only non-trivial observation is that $\sigma(v_i) \neq \sigma(v_j)$ for all $(v_i, v_j) \in E$, hence every triple in $\{C_\alpha\ e\ C_\beta \mid (v_\alpha, v_\beta) \in E\}$ is mapped into G_{e2} by μ'.

To conclude the proof, finally note that G trivially satisfies $\mathcal{C}$.

C.2. Proof of Lemma 5.7

First it remains to prove the two claims given as part of the explanation of the intuition in Section 5.

Claim 1: Consider G, and q as defined by the reduction together with the following sets: $\vec{X} = \{(x_1, x_2) \mid x_1, x_2 \in \{0, 1, 2\}, x_1 \neq x_2\}$, $\vec{X}^* = \{(x_1^*, x_2^*) \mid x_1^*, x_2^* \in \{0, 1, 2\}\}$, and $C = \{0, 1, 2\}^p$. Then $q(G) = \vec{X} \times \vec{X}^* \times C$.

Proof of Claim 1: The claim follows immediately under the assumption that E is mapped to e^*.

Claim 2: Let $G' \subset G$ such that $q(G') = q(G)$. Then $G' = G \setminus \{e^*\ s\ e^*\}$.

Proof of Claim 2: It is easy to see that if any other triple is removed from G, then $q(G')$ cannot contain all answers described in Claim 1: If $G_{cols} \nsubseteq G'$, then some values on the C_i are missing. If $G_{e1} \cup G_{e2} \nsubseteq G'$ then some value on (X_1, X_2) or (X_1^*, X_2^*) cannot be retrieved over G'. Finally, if $\{e\ s\ e\} \nsubseteq G'$, then $q(G') = \emptyset$.

Finally it remains to show that $(G, \emptyset, \emptyset, \mathcal{Q})$ is a positive instance of MINI-RDF$^{\subseteq, CQ}$ iff $\mathcal{G}$ is a positive instance of Q-3COL$_{2,\forall}$. The two directions are shown separately. In the following, colorings on nodes $v_i \in V$ are identified with the URIs $0, 1, 2$.

"If"-direction Assume that for every possible coloring σ_1 on V_1, there exists a coloring σ_2 on V_2 such that $\sigma = \sigma_1 \cup \sigma_2$ is a valid 3-coloring of (V, E). Then define $G' = G \setminus \{e^*\ s\ e^*\}$, and let $(\mu(X_1), \mu(X_2), \mu(X_1^*), \mu(X_2^*),\ \mu(C_{i_1}), \dots, \mu(C_{i_p}))$ be an arbitrary tuple in $q(G)$. It must be shown that it is also in $q(G')$. Towards this goal, define a coloring σ_1 on V_1 as $\sigma_1(v_i) = \mu(C_i)$ for every $v_i \in V_1$ (hence $C_i \in \{C_{i_1}, \dots, C_{i_p}\}$). Next let σ_2 be a coloring on V_2 such that $\sigma_1 \cup \sigma_2$ is a valid 3-coloring on (V, E). Then $\mu'(\mathit{body}(q)) \subseteq G'$ holds for the extension μ' of μ defined as $\mu'(E) = e$ and $\mu'(C_i) = \sigma_2(v_i)$ for all $v_i \in V_2$. To see this, just note that $\sigma(v_i) \neq \sigma(v_j)$ for all $(v_i, v_j) \in E$. Hence $\mu'(C_\alpha) \neq \mu'(C_\beta)$ for every triple in $\{C_\alpha\ E\ C_\beta \mid (v_\alpha, v_\beta) \in E\}$. Therefore this part of $\mathit{body}(q)$ is mapped into G_{e2} by μ'. From this the result follows easily.

"Only if"-direction Assume that $q(G') = q(G)$ for $G' = G \setminus \{e^*\ s\ e^*\}$, and consider an arbitrary coloring σ_1 on V_1. It follows from Claim 1 that there exists a homomorphism $\mu\colon \mathit{body}(q) \to G'$ s.t. $\mu(C_i) = \sigma_1(v_i)$ for all $v_i \in V_1$. Since $\{e^*\ s\ e^*\} \nsubseteq G'$, $\mu(E) = e$. Hence because of $\{C_\alpha\ E\ C_\beta \mid (v_\alpha, v_\beta) \in E\} \subseteq \mathit{body}(q)$ and $\mu(E) = e$, it must be the case that $\mu(C_\alpha) \neq \mu(C_\beta)$ for all $(v_\alpha, v_\beta) \in E$. It can therefore be easily seen that $\sigma = \sigma_1 \cup \sigma_2$ with $\sigma_2(v_i) = \mu(C_i)$ for all $v_i \in V_2$ is a valid 3-coloring on (V, E). This concludes the proof.

C.3. Proof of Lemma 5.9

As a first step, the two claims presented in Section 5 are shown.

Claim 1: Let $G' \subset G$ such that $q(G) = q(G')$ for every $q \in \mathcal{Q}$. Then $G' = G \setminus \{0\ e^*\ 0\}$.

Proof of Claim 1: This observation follows immediately from the query $q\colon \{Y_1\ e\ Y_2 \,.\, Y_1\ e^*\ Y_2\} \to \mathit{ans}(Y_1, Y_2)\}$ in $\mathcal{Q}$, since removing one triple from $G_{e1} \cup G_{e2}$ would decrease the number of results in $q(G')$.

Claim 2: Let m be the size of the biggest clique in (V, E). If m is odd, then $q_i(G) = \{(o_{i+1})\}$ for $i \in \{2, 4, \dots, m-1\}$ and $q_i(G) = \emptyset$ for $i \in \{m+1, m+3, \dots, n\}$. If m is even, then $q_i(G) = \{(o_{i+1})\}$ for

$i \in \{2, 4, \ldots, m\}$ and $q_i(G) = \emptyset$ for $i \in \{m+2, m+4, \ldots, n\}$.

Proof of Claim 2: First of all, note that mapping $X'_1, \ldots, X'_{i+1}$ to 0 maps every triple $X'_r\ e^*\ X'_s$ into G_{e2}. Hence $q_i(G) = \{(o_{i+1})\}$ if the triples in $body(q_i)$ containing $X_1, \ldots, X_i$ can be mapped to G_{e1}, and $q_i(G) = \emptyset$ otherwise. Now it is easy to see that G_{e1} encodes the edges in E, and that $q_i(G) \neq \emptyset$ iff (V, E) contains a clique of size i.

Using these claims, the correctness of the reduction can be easily shown: Assume that the biggest clique in (V, E) is indeed odd, say m. Hence by Claim 2, $q_{m-1}(G) \neq \emptyset$ but $q_{m+1}(G) = \emptyset$. To see that this holds for $G' = G \setminus \{0\ e^*\ 0\}$ as well, note that G_{e2} encodes a clique of size m. Hence there exists a mapping $\tau\colon \{X'_1, \ldots, X'_m\} \cup \{X_1, \ldots, X_{m-1}\} \to \{v_1, \ldots, v_n\}$ such that $\tau(body(q_i)) \subseteq G'$. Obviously, the result of all other queries remains unchanged as well.

Next, assume that $q(G') = q(G)$ for every $q \in \mathcal{Q}$. Then by Claim 1, $G' = G \setminus \{0\ e^*\ 0\}$. Further, assume to the contrary that the biggest clique in (V, E) is even, say m. Then $q_m(G) \neq \emptyset$. However, it can be easily checked that $q_m(G') = \emptyset$. This gives a contradiction and therefore concludes the proof.

C.4. Proof of Lemma 5.10

In Section 5 it was left open to prove the two claims stating the two major properties of the reduction. These proofs are given first. Then, using these properties, the correctness of the reduction is shown.

Claim 1: Let m be the size of the biggest clique in (V, E). Then $Cl_{\mathcal{R}}(G) = G \cup \{i\ clique\ i \mid 2 \leq i \leq m\}$.

Proof of Claim 1: It can be easily verified that over G, each rule r_i (for $i \in \{2, \ldots, n\}$) is applicable iff (V, E) contains a clique of size i, since this allows to find a homomorphism from $\{X_r\ e\ X_s \mid 1 \leq r < s \leq i\}$ into G_e.

Claim 2: Assume $G' \subset G$ such that $q(Cl_{\mathcal{R}}(G')) = q(Cl_{\mathcal{R}}(G))$. Then $G \setminus G' = Cl_{\mathcal{R}}(G) \setminus Cl_{\mathcal{R}}(G') = \{c\ c\ c\}$.

Proof of Claim 2: It can be easily verified that $q(Cl_{\mathcal{R}}(G')) = q(Cl_{\mathcal{R}}(G))$ cannot hold if a single triple from $G_{ord} \cup G_e \cup \{i\ clique\ i \mid 2 \leq i \leq m\}$ (where m is the size of the biggest clique) is not in $Cl_{\mathcal{R}}(G')$. From this it follows immediately that $G_{ord} \cup G_e = G'$.

To prove the correctness of the reduction, first assume that the size of the biggest clique is odd, say m. Then let $G' = G \setminus \{c\ c\ c\}$, and $s = (\mu(X), \mu(Y), \mu(V_1), \mu(V_2), \mu(C)) \in q(Cl_{\mathcal{R}}(G))$. If $\mu(C)$ is odd, then obviously $\mu(C)\ clique\ \mu(C)$ is still contained in $Cl_{\mathcal{R}}(G')$, hence $s \in q(Cl_{\mathcal{R}}(G'))$. On the other hand, if $\mu(C)$ is even, then $\mu(C)\ clique\ \mu(C)$ can no longer be derived from G' via $r_{\mu(C)}$. However, since the size of the biggest clique in (V, E) is odd, there exists a clique of size $\mu(C) + 1$ in (V, E). Therefore the triple $\mu(C) + 1\ clique\ \mu(C) + 1$ is in $Cl_{\mathcal{R}}(G')$. Since G' further contains the triple $\mu(C)\ succ\ \mu(C) + 1$, the triple $\mu(C)\ clique\ \mu(C)$ can be obtained by the rule $\{X\ succ\ Y . Y\ clique\ Y\} \Rightarrow \{X\ clique\ X\}$. Therefore $s \in q(Cl_{\mathcal{R}}(G'))$.

Now assume that $q(Cl_{\mathcal{R}}(G')) = q(Cl_{\mathcal{R}}(G))$ for $G' = G \setminus \{c\ c\ c\}$, and consider $s = (\mu(X), \mu(Y), \mu(V_1), \mu(V_2), \mu(C)) \in q(Cl_{\mathcal{R}}(G))$ such that $\mu(C) = m$ takes the value of the size of the largest clique in (V, E). Then $\mu(C)$ is odd: Assume to the contrary that $\mu(C)$ is even. Then, since r_m for even m is not applicable on G', the only way for the triple $m\ \ clique\ \ m$ to be in $Cl_{\mathcal{R}}(G')$ is because of some triple $m + 1\ \ clique\ \ m + 1$ and the rule $\{X\ \ succ\ Y . Y\ clique\ Y\} \Rightarrow \{X\ clique\ X\}$ in $\mathcal{R}$. Since $m + 1\ clique\ m + 1$ is in $Cl_{\mathcal{R}}(G')$ iff (V, E) contains a clique of size $m + 1$, this gives the desired contradiction.

C.5. Proof of Lemma 5.15

The proof is by reduction from Q-3COL$_{\forall,2}$. Hence let $\mathcal{G} = ((V, E), (V_1, V_2))$ be an arbitrary instance of Q-3COL$_{\forall,2}$ with $V = \{v_1, \ldots, v_n\}$ and $|V_1| = p$. Define an instance $(G, \mathcal{R}, \mathcal{Q})$ of RDF-RULEMIN$^{\subseteq, CQ}(G, \mathcal{R}, \mathcal{Q})$ as follows. Let $G = G_{cols} \cup G_e$ where

$$G_{cols} = \{0\ iscol\ 0 . 1\ iscol\ 1 . 2\ iscol\ 2\}, \text{ and}$$
$$G_e \quad = \{0\ e\ 2 . 0\ e\ 1 . 1\ e\ 2 . 1\ e\ 0 . 2\ e\ 0 . 2\ e\ 1\}.$$

Further, $\mathcal{R}$ contains a single (fixed) rule r

$$\mathcal{R} = \{\{X\ iscol\ X\} \Rightarrow \{X\ e\ X\}\}$$

and $\mathcal{Q}$ contains the single query q

$$\begin{aligned}\mathcal{Q} = \{\ &\{C_i\ iscol\ C_i \mid v_i \in V\} \cup \\ &\quad \{C_i\ e\ C_j \mid \{v_i, v_j\} \in E\} \\ &\qquad\qquad \to ans(C_{i_1}, \ldots, C_{i_p})\ \}\end{aligned}$$

where $C_1, \ldots, C_n$ are new variables, one for each $v_i \in V$ and $C_{i_1}, \ldots, C_{i_p}$ are those $C_i \in \{C_1, \ldots, C_n\}$ such that $v_i \in V_1$. Obviously, this reduction is feasible in LOGSPACE. The crucial observation towards the correctness of the reduction is formalised in Claim 1 and follows immediately from the obvious fact that $\{0\ e\ 0\,.\,1\ e\ 1\,.\,2\ e\ 2\} \subseteq Cl_{\mathcal{R}}(G)$.

Claim 1: $q(Cl_{\mathcal{R}}(G)) = \{0, 1, 2\}^p$.

Hence let $s \in \{0, 1, 2\}^p$. Then $s \in q(Cl_{\emptyset}(G)) = q(G)$ iff $\mathcal{G}$ is a positive instance of Q-3COL$_{\forall,2}$. This can be shown by establishing a one-to-one correspondence between coloring on V_1 and mappings $\{C_{i_1}, \ldots, C_{i_p}\} \to \{0, 1, 2\}$ on the one hand and between colorings on V_2 and mappings $\{C_1, \ldots, C_n\} \setminus \{C_{i_1}, \ldots, C_{i_p}\} \to \{0, 1, 2\}$ on the other hand. Since the proof is very similar to the proof of Lemma 5.7, its details are omitted.

D. Full proofs of Section 6

D.1. Proof of Theorem 6.1

In the proof of Theorem 6.1 it was left open to show that the presented reduction is indeed correct, i.e. that there exists a vertex cover of size k' iff there exists a subgraph G' containing $3 * m + 2 + k' + |V|$ triples, s.t. $G^{rdf} \subseteq Cl_{\mathcal{R}}(G')$ and $|G'| = k$. Before showing the two directions separately, first consider the size of G^{rdf}, which is $2 * m + |V| + |V| + 2 + m$, where $m = |E|$. This number is derived as follows: Because every edge connects two different nodes (recall that G is assumed to contain no selfloops), G^{rdf} contains $2 * m$ triples of the form $\{v_i\ neighbour\ e_k\}$. It further contains $|V|$ triples of the form $\{v_i\ backupv\ v_i\}$, $|V|$ triples of the form $\{v_i\ \ v\ \ v_i\}$, the two triples $\{e_m\ last\ e_m\,.\,e_0\ in\ e_0\}$, and m triples $\{e_i\ succ\ e_{i+1}\}$ (for $i \in \{0, \ldots, m-1\}$).

"If" direction) Assume that such a G' exists. It must be shown that then there exists a vertex cover of size k'. First note that the only triples that can be removed from G^{rdf} such that $G^{rdf} \subseteq Cl_{\mathcal{R}}(G')$ holds are triples of the form $v_i\ \ v\ \ v_i$. Denoting the number of such triples that remain in G' with r, the size of G' is obviously $2 * m + |V| + 2 + m + r$. (I.e. G' differs by $|V| - r$ triples of the form $v_i\ v\ v_i$ from G^{rdf}.)

Now define a vertex cover of G to contain exactly those nodes whose corresponding triples $v_i\ v\ v_i$ are in G', i.e. $VC = \{v_i \mid v_i\ v\ v_i \in G'\}$. As by assumption $|G'| = k = 3*m+2+k'+|V|$, it follows that $k' = r$, hence VC has the required size. It remains to show that VC is indeed a valid vertex cover: Because G' satisfies $G^{rdf} \subseteq Cl_{\mathcal{R}}(G')$, the third rule is applicable, hence $e_m\ in\ e_m \in Cl_{\mathcal{R}}(G')$ (remember the assumed arbitrary ordering on E). By induction along this order, it is easy to check that $e_j\ in\ e_j \in Cl_{\mathcal{R}}(G')$ only holds if VC covers all edges $e_\ell \leq e_j$ (again $\leq$ with respect to the the ordering on E). To see this, just note that $e_\ell\ in\ e_\ell \in Cl_{\mathcal{R}}(G')$ iff $\{e_{\ell-1}\ in\ e_{\ell-1}\,.\,e_\ell\ e\ e_\ell\} \subset Cl_{\mathcal{R}}(G')$, and that for every e_ℓ the triple $e_\ell\ e\ e_\ell$ is in $Cl_{\mathcal{R}}(G')$ only if for a vertex v_i adjacent to e_ℓ, $v_i\ v\ v_i$ is in G'. Therefore, since $\{e_m\ in\ e_m\} \subset Cl_{\mathcal{R}}(G')$ it follows that VC indeed covers all $e_j \in E$.

"Only-if" direction) Assume that there exists a Vertex Cover $VC \subseteq V$ with $|VC| \leq k'$. It must be shown that then there exists a subgraph G' of size k such that $G^{rdf} \subseteq Cl_{\mathcal{R}}(G')$ holds (as $\mathcal{C} = \emptyset$ is trivially satisfied). Towards this goal, let $G' = G^{rdf} \setminus \{v_i\ v\ v_i \mid v_i \in V \setminus VC\}$. Now the claim is that G' is the desired subgraph: Since VC is a valid vertex cover, for every edge at least one of its endpoints lies in VC, say v_i. Hence $v_i\ v\ v_i \in G'$. Therefore $\{e_\ell\ e\ e_\ell \mid e_\ell \in E\} \subset Cl_{\mathcal{R}}(G')$, by the first rule. Because of this, also $\{e_\ell\ in\ e_\ell \mid e_\ell \in E\} \subset Cl_{\mathcal{R}}(G')$, which finally allows us to conclude $\{v_i\ v\ v_i \mid v_i \in V\} \subset Cl_{\mathcal{R}}(G')$, which proves the case. The size $|G'| = 3m + 2 + |V| + k'$ follows trivially from the definition of G' and the size of G^{rdf}.

D.2. Proof of Theorem 6.2

Membership follows immediately from Theorem 3.1.

Hardness is shown by an appropriate adaption of the proof of Lemma 3.5. That is, let $\mathcal{G} = ((V, E), (V_1, V_2, V_3))$ be an arbitrary instance of Q-3COL$_{\exists,3}$ with $V = \{v_1, \ldots, v_n\}$. Define an RDF graph G and a set $\mathcal{C}$ of constraints as follows. Let $G = G_{cols} \cup G_{v1} \cup G_{col1} \cup G_{col2} \cup G_{e1} \cup G_{e2} \cup G_{neq}$ be defined as in the proof of Lemma 3.5, and $\mathcal{C}$ be defined as $\mathcal{C} = \mathcal{C}_{basic} \cup \mathcal{C}_{col1} \cup \mathcal{C}_0 \cup \mathcal{C}_{\mathcal{G}}$ where

$$\begin{aligned}\mathcal{C}_{basic} &= \{\{S\ P\ O\} \Rightarrow G_{cols} \cup G_{v1} \cup G_{col2} \cup \\ &\qquad\qquad G_{e2} \cup G_{neq}\},\\ \mathcal{C}_{col1} &= \{\{X\ v\ X\} \Rightarrow \{C\ iscol\ C\,.\,X\ a\ C\};\\ &\quad \{X\ a\ C_1\,.\,X\ a\ C_2\,.\,C_1\ neq\ C_2\}\\ &\qquad\qquad \Rightarrow \{0\ e\ 0\}\},\end{aligned}$$

and $\mathcal{C}_0$ and $\mathcal{C}_{\mathcal{G}}$ are defined as in the proof of Lemma 3.5. Again, introducing new URIs v_i for each $v_i \in V$, by slight abuse of notation v_i is used to denote both, vertices in V and URIs in G.

Obviously, the reduction is feasible in LOGSPACE. The idea of the reduction is almost the same as that of the reduction used to prove Lemma 3.5. The only difference is that in the aforementioned proof the triples that must not be removed from G were defined implicitly by not providing rules to recover them. In contrast, now all these triples are explicitly required to remain in G by $\mathcal{C}_{basic} \cup \mathcal{C}_{col1}$.

Due to the close similarity to Lemma 3.5, the details of the correctness proof are omitted, apart from the following two notes:

- If $\mathcal{G}$ is a positive instance of Q-3COL$_{\exists,3}$, then a subgraph $G' \subset G$ that satisfies $\mathcal{C}$ is defined the same way as described in the "only if"-direction in the proof of Lemma 3.5.
- Let $G' \subset G$ such that G' satisfies G. Then it satisfies the properties (i) – (iii) stated in the "if"-direction in the proof of Lemma 3.5.

D.3. Proof of Theorem 6.3

Membership can be established by the same algorithm as in the proof of Lemma 3.4: Recall the algorithm presented there, and note that in step 2, for every constraint $c \in \mathcal{C}$ a call to a Π_2^P-oracle was used to check if subgraph G' still satisfies c. This test was done by verifying that for every homomorphism $h\colon \mathit{body}(c) \to G'$ there exists an extension h' of h that is a homomorphism $h'\colon \mathit{head}(c) \to G'$. Now replacing tgds by full tgds, this check no longer requires an Π_2^P oracle, but a coNP-oracle suffices: One only has to check for every $c \in \mathcal{C}$ if every homomorphism $h\colon \mathit{body}(c) \to G'$ is also a homomorphism $\mathit{head}(c) \to G'$.

Hardness is shown for the second case described in the theorem, i.e. for the problem of deciding if there exists some $G' \subset G$ with $G' \neq \emptyset$ such that G' satisfies $\mathcal{C}$. The proof is by reduction from Q-3COL$_{\exists,2}$. Hence let $\mathcal{G} = ((V,E),(V_1,V_2))$ be an arbitrary instance of Q-3COL$_{\exists,2}$, with $V = \{v_1,\ldots,v_n\}$. Define an instance $(G,\mathcal{C})$ of the above problem as follows. Let $G = G_{cols} \cup G_{v1} \cup G_{col1} \cup G_{col2} \cup G_{e1} \cup G_{e2} \cup G_{neg} \cup G_c$ where

$$\begin{aligned}
G_{cols} &= \{0\ \mathit{iscol}\ 0\,.\,1\ \mathit{iscol}\ 1\,.\,2\ \mathit{iscol}\ 2\},\\
G_{v1} &= \{v_i\ v\ v_i \mid v_i \in V_1\},\\
G_{col1} &= \{v_i\ a\ 0\,.\,v_i\ a\ 1\,.\,v_i\ a\ 2 \mid v_i \in V_1\},\\
G_{col2} &= \{v_i\ b\ 0\,.\,v_i\ b\ 1\,.\,v_i\ b\ 2 \mid v_i \in V_2\},\\
G_{e1} &= \{0\ e\ 0\,.\,1\ e\ 1\,.\,2\ e\ 2\},\\
G_{e2} &= \{0\ e\ 2\,.\,0\ e\ 1\,.\,2\ e\ 0\,.\,2\ e\ 1\,.\,1\ e\ 2\,.\,1\ e\ 0\},\\
G_{neq} &= \{0\ \mathit{neq}\ 2\,.\,0\ \mathit{neq}\ 1\,.\,2\ \mathit{neq}\ 0\,.\,2\ \mathit{neq}\ 1\,.\\
&\qquad 1\ \mathit{neq}\ 2\,.\,1\ \mathit{neq}\ 0\},
\end{aligned}$$

and every v_i is a new URI for some $v_i \in V$ (by slight abuse of notation, let v_i denote both, nodes $v_i \in V$ and URIs in G). Finally, the set $\mathcal{C}$ of constraints is defined as $\mathcal{C} = \mathcal{C}_{basic} \cup \mathcal{C}_{v1} \cup \mathcal{C}_{col1} \cup \mathcal{C}_0 \cup \mathcal{C}_{\mathcal{G}}$ where

$$\begin{aligned}
\mathcal{C}_{basic} &= \{\{S\ P\ O\}\\
&\qquad \Rightarrow G_{cols} \cup G_{v1} \cup G_{col2} \cup G_{e2} \cup G_{neq}\},\\
\mathcal{C}_{v1} &= \{\{X\ v\ X\} \Rightarrow \{C\ \mathit{iscol}\ C\,.\,X\ a\ C\}\},\\
\mathcal{C}_{col1} &= \{\{X\ a\ C\,.\,X\ a\ D\,.\,C\ \mathit{neq}\ D\} \Rightarrow \{0\ e\ 0\},\\
\mathcal{C}_0 &= \{\{X\ e\ X\,.\,Y\ \mathit{iscol}\ Y\} \Rightarrow \{Y\ e\ Y\};\\
&\qquad \{Z\ e\ Z\,.\,X\ v\ X\,.\,Y\ \mathit{iscol}\ Y\} \Rightarrow \{X\ a\ Y\}\}\\
\mathcal{C}_{\mathcal{G}} &= \{\{v_i\ a\ C_i \mid v_i \in V_1\} \cup \{v_i\ b\ C_i \mid v_i \in V_2\} \cup\\
&\qquad \{C_\alpha\ e\ C_\beta \mid (e_i,e_j) \in E, \alpha = \min(i,j),\\
&\qquad \beta = \max(i,j)\} \Rightarrow \{0\ e\ 0\}\}.
\end{aligned}$$

Obviously, this reduction is feasible in LOGSPACE. Towards both, its correctness and intuition, consider the following claim, which is left without proof.

Claim 1: Let $G' \subset G$ with $\emptyset \neq G'$ and such that G' satisfies $\mathcal{C}$. Then $\mathcal{G}_{cols} \cup G_{v1} \cup G_{col2} \cup G_{e2} \cup G_{neq} \subseteq G'$, and $G' \cap G_{e1} = \emptyset$. Further, $|G^i_{col1} \cap G'| = 1$ where G^i_{col1} is as defined in the proof of Lemma 3.5.

Since the correctness proof is analogous to that of Lemma 3.5, it is omitted. Instead only some important properties are stated:

- If $\mathcal{G}$ is a positive instance of Q-3COL$_{\exists,2}$, then $G' \subset G$ can be defined from the coloring σ_1 on V_1 as $G' = \mathcal{G}_{cols} \cup G_{v1} \cup G_{col2} \cup G_{e2} \cup G_{neq} \cup \{v_i\ a\ \sigma_1(v_i) \mid v_i \in V_1\}$
- If $G' \subset G$ with $\emptyset \neq G'$ and such that G' satisfies $\mathcal{C}$, then
  - $*$ $G_{col1} \cap G'$ encodes a coloring σ_1 on V_1.
  - $*$ Every homomorphism $\tau\colon \mathit{body}(c)\backslash\{C_\alpha\ e\ C_\beta \mid (v_\alpha, v_\beta) \in E\} \to G'$ (where c is the only tgd in $\mathcal{C}_{\mathcal{G}}$) encodes a coloring σ on V such that $\sigma(v_i) = \sigma_1(v_i)$ for all $v_i \in V_1$.
  - $*$ Since $0\ e\ 0 \notin G'$, the only way to satisfy c is that none of the homomorphisms τ from above also maps $\{C_\alpha\ e\ C_\beta \mid (v_\alpha, v_\beta) \in E\}$ into G'.
  - $*$ This is exactly the case if none of these homomorphisms encodes a valid 3-coloring of V.

D.4. Proof of Theorem 6.4

Membership follows immediately from Theorem 3.1. *Hardness* can be shown by adapting the reduction presented in the proof of Lemma 3.7, similar as the adaption of the proof of Lemma 3.5 above. Recall the reduction from 3COL used to prove Lemma 3.7. Given an instance (V,E) of 3COL, let $(G,\mathcal{R},\mathcal{C})$ be the instance of MINI-RDF$^{\subseteq}$ as defined in the proof of Lemma 3.7. An instance $(\hat{G},\hat{\mathcal{C}})$ of the problem under consideration is then defined as follows: $\hat{G} = G \cup \{v_i\ v\ v_i \mid v_i \in V_1\}$, and $\hat{\mathcal{C}} = \mathcal{C} \cup \mathcal{C}_{basic} \cup \mathcal{C}_{v1}$ where

$\mathcal{C}_{basic} = \{\{S\ P\ O\} \Rightarrow G_{cols} \cup G_e \cup G_{neq}\},$
$\mathcal{C}_{v1} \quad = \{\{X\ v\ X\} \Rightarrow \{Y\ iscol\ Y\ .\ X\ a\ Y\}\}$

Obviously, this reduction is feasible in LOGSPACE. The idea of the reduction remains unchanged, only that instead of expressing implicitly which triples must be contained in every "valid" $G' \subset G$ (by not defining rules to derive them), these triples are now explicitly listed by $\mathcal{C}_{basic}$ and $\mathcal{C}_{v1}$. Due to the close similarity to the proof of Lemma 3.7, the details of the correctness proof are omitted, apart from the following two notes:

- Given a valid 3-coloring of (V, E), a subgraph $G' \subset \hat{G}$ that satisfies $\hat{\mathcal{C}}$ can be derived just as defined in the "if"-direction of the proof of Lemma 3.7.
- Let $G' \subset \hat{G}$ that satisfies $\hat{\mathcal{C}}$. Then G' satisfies the properties (i) – (iv) stated in the "only-if" direction of the proof of Lemma 3.7.